

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO ĐỒ ÁN

Chủ đề 08

Khảo sát Inpainting và ứng dụng vào loại bỏ đối tượng

Môn học: Xử lý ảnh số và video số

Giảng viên hướng dẫn

Lý Quốc Ngọc

Nhóm Ngọt

MSSV Thành viên

21120180 Nguyễn Bích Khuê (NT)

21120161 Tiêu Ân Tuấn

21120291 Nguyễn Đức Nam

Thành phố Hồ Chí Minh, 2023

Mục lục

	Trang
Mục Lục	2
Danh sách Bảng.....	3
Danh sách Hình.....	4
CHAPTER I: Giới thiệu.....	5
1. Ý nghĩa chủ đề	5
1.1 Ý nghĩa khoa học	5
1.2 Ý nghĩa ứng dụng của chủ đề	5
1.3 Phát biểu bài toán:	6
1.4 Đóng góp:	9
CHAPTER II: Các công trình nghiên cứu liên quan	10
1. Phương pháp truyền thống	10
1.1 Giới thiệu về phương pháp Exemplar-based	10
1.2 Giới thiệu về phương pháp Diffusion-Based	11
2. Phương pháp Deep Learning	14
2.1 Encoder Decoder	14
2.2 CNN - Convolutional Neural Network	15
2.3 GANs - Generative Adversarial Networks	17
2.4 Nhận xét	18
CHAPTER III: Phương pháp Lama	20
1. Dữ liệu huấn luyện.....	20
1.1 Thuật toán Mask Generation	20
1.2 Places Dataset	22
2. Kiến trúc mạng.....	23
2.1 Đặt vấn đề	23
2.2 Tổng quát	24
2.3 Fast Fourier Convolution	24
3. Loss function.....	26
3.1 Perceptual Loss	26
3.2 Adversarial loss	26

CHAPTER IV: Kết quả thử nghiệm.....	29
1. Phần code.....	29
2. Kết quả.....	30
2.1 Kết quả 1	30
2.2 Kết quả 2	30
CHAPTER V: Kết luận và hướng phát triển.....	32
1. Kết luận	32
2. Hướng phát triển.....	33

Danh sách bảng

Danh sách hình vẽ

I.1	Ứng dụng giúp hồi phục lại các ảnh bị trầy xước, hỏng	5
I.2	Ứng dụng giúp loại bỏ đối tượng trong chỉnh sửa ảnh	6
I.3	Ứng dụng trong image-based rendering (IBR)	6
I.4	Ảnh đầu vào bị hỏng	7
I.5	Mặt nạ có cùng kích thước với ảnh đầu vào	7
I.6	Ảnh đầu ra	8
I.7	So sánh ảnh gốc và ảnh đầu ra	8
II.1	Phương pháp tổng hợp texture dựa trên Pixel	11
II.2	Minh họa phương pháp Diffusion Based	11
II.3	Mô hình encoder - decoder	14
II.4	Ví dụ Encoder - Coder training [9]	15
II.5	Minh họa mô hình CNN.	16
II.6	Minh họa mô hình CNN trong image inpainting.	16
II.7	Minh họa mạng GANs đơn giản.	17
II.8	Minh họa một loại loss function cho mô hình GANs.	18
III.1	Phần màu đỏ: các đối tượng trích xuất được từ công cụ Detectron2, Phần màu trắng: Segm mask lấy được bằng việc trích xuất bóng của đối tượng từ ảnh khác	22
III.2	Ví dụ của Wide Masks và Box Masks với $p_{irr} = 0.5$	23
III.3	Tham số của mặt nạ được tạo với các độ rộng: hẹp (narrow), trung bình (Medium) và rộng (Wide)	23
III.4	Kiến trúc mạng Lama sử dụng FFC	24
III.5	Fast Fourier Convolution	25
III.6	Công thức mô tả đầu vào và đầu ra của khối FFC	25
III.7	Perceptual Loss	26
III.8	Adversarial Loss 1	27
III.9	Final Loss function	27
IV.1	Input đầu vào 1	30
IV.2	Output1	30
IV.3	Input đầu vào 2	30
IV.4	Output2	31

Chapter I

Giới thiệu

1. Ý nghĩa chủ đề

1.1 Ý nghĩa khoa học

- Image Inpainting là quá trình giúp hồi phục các pixel hoặc các vùng bị hỏng và bị thiếu của một bức ảnh.
- Image Inpainting giúp tái tạo lại thông tin của ảnh để giảm thiểu sự mất mát thông tin quan trọng

1.2 Ý nghĩa ứng dụng của chủ đề

- Giúp hồi phục lại các ảnh bị trầy xước, hỏng



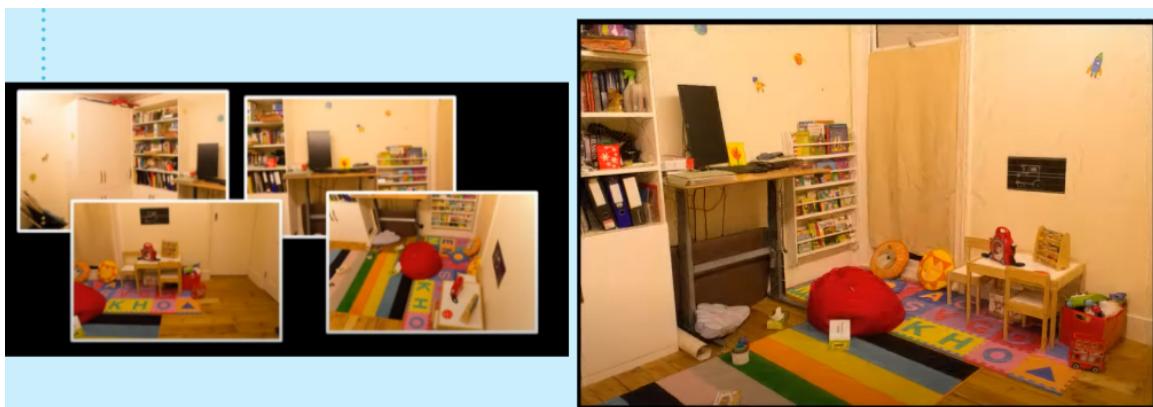
Hình I.1: Ứng dụng giúp hồi phục lại các ảnh bị trầy xước, hỏng

- Giúp loại bỏ đối tượng trong chỉnh sửa ảnh



Hình I.2: Ứng dụng giúp loại bỏ đối tượng trong chỉnh sửa ảnh

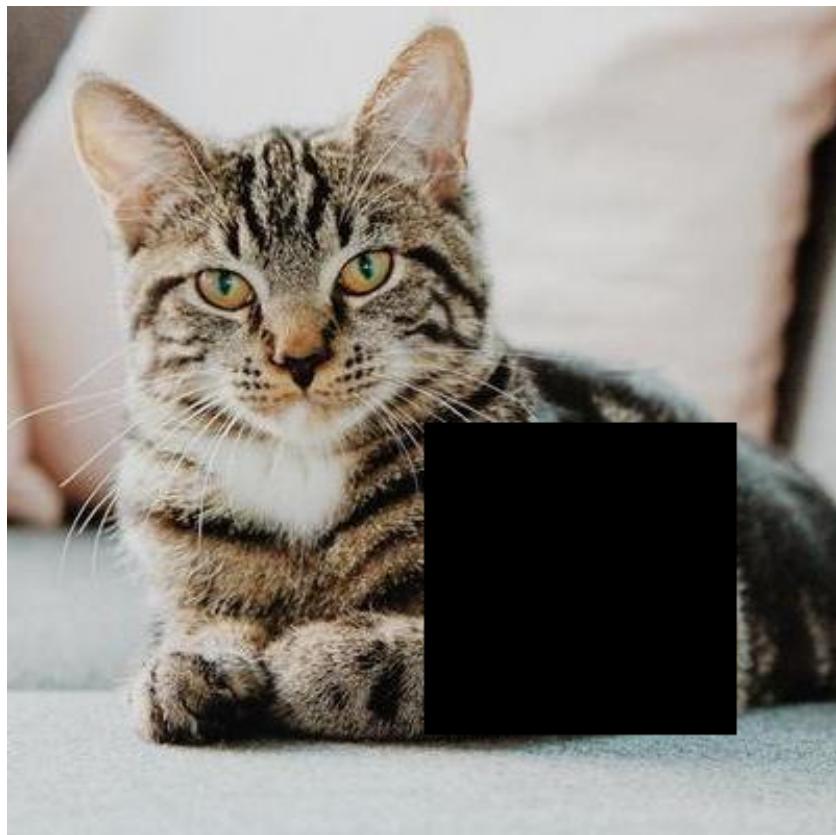
- Dóng vai trò quan trọng trong việc điền vào những phần thiếu sót trong hình ảnh, những vùng bị che khuất trong image-based rendering (IBR)



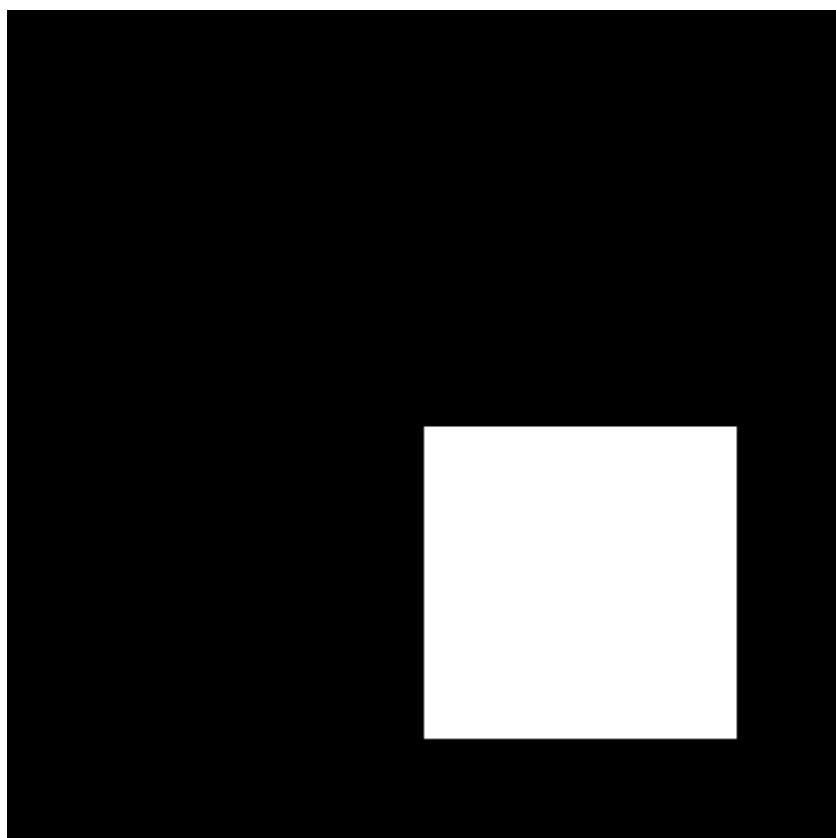
Hình I.3: Ứng dụng trong image-based rendering (IBR)

1.3 Phát biểu bài toán:

- Input:
 - Một bức ảnh cần khôi phục có kích thước $n \times m$
 - Mặt nạ có cùng kích thước với ảnh ($n \times m$): Là mảng chứa các giá trị 0 hoặc 1 (với 0 là vị trí pixel nguyên vẹn và 1 vị trí pixel cần khôi phục). Một số phương pháp có thể không cần mặt nạ nên đầu vào mặt nạ sẽ là một tùy chọn.
- Ví dụ:



Hình I.4: Ảnh đầu vào bị hỏng



Hình I.5: Mặt nạ có cùng kích thước với ảnh đầu vào

- Output: Ảnh đã được khôi phục các pixel bị hỏng hoặc mất mát.



Hình I.6: Ảnh đầu ra

- Ảnh số: Giá trị pixel cần khôi phục lấy từ đâu ra?

Mục đích của Image Inpainting là chỉ cố gắng hồi phục ảnh sao cho tự nhiên nhất và phù hợp với ngữ cảnh nhất chứ không thể hồi phục lại 100% ảnh gốc, chúng ta có thể so sánh bức ảnh ảnh gốc và ảnh đầu ra của ví dụ vừa rồi



Hình I.7: So sánh ảnh gốc và ảnh đầu ra

Framework chung:

- Bước 1: Truyền vào bức ảnh bị hỏng và mặt nạ có cùng kích thước (mặt nạ tùy chọn)
- Bước 2: Sử dụng các phương pháp Inpainting để tính toán giá trị và điền vào các pixel bị thiếu và đây chính là bước quan trọng nhất
- Bước 3: Trả về hình ảnh đã được khôi phục các pixel bị hỏng hoặc mất mát.

1.4 Đóng góp:

- Báo cáo này sẽ giới thiệu sơ lược về Image Inpainting và một số phương pháp được sử dụng trong Image Inpainting
- Nhóm sẽ tập trung vào nghiên cứu và trình bày ứng dụng Inpainting trong Remove Object

Chapter II

Các công trình nghiên cứu liên quan

1. Phương pháp truyền thống

1.1 Giới thiệu về phương pháp Exemplar-based

Nguyên lý

Tìm kiếm bản vá giống nhất [4] [6] trong phần đã biết của ảnh cần được inpainting với vùng lân cận đã biết của khu vực chưa biết cần điền vào của ảnh đầu vào, sau đó sao chép pixel trung tâm hoặc một tập hợp các pixel từ bản vá vào vị trí đó. Điều này giúp khôi phục thông tin từ phần đã biết sang phần không biết của hình ảnh, tạo ra một hình ảnh hoàn chỉnh.

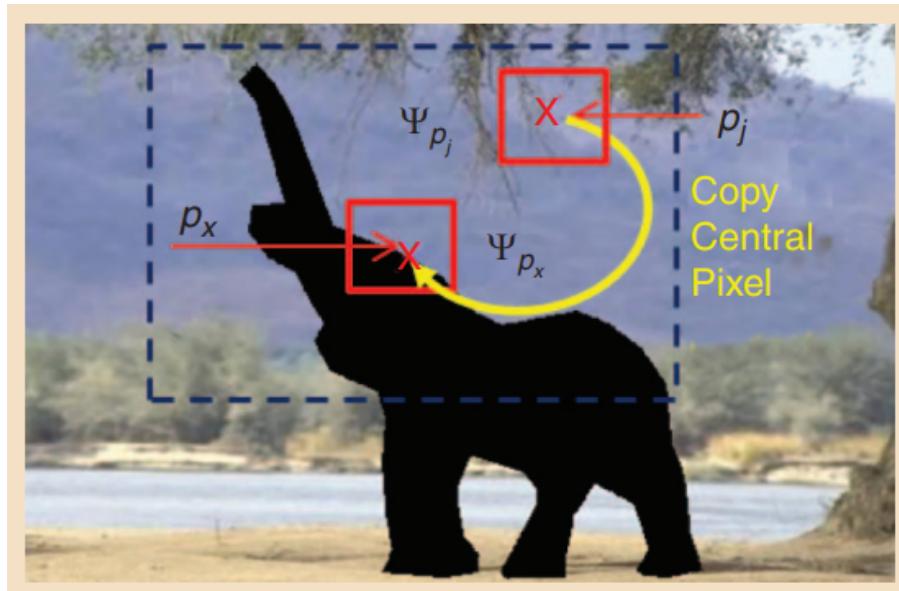
Thuật toán

Một kỹ thuật đơn giản nhất dựa trên pixel như sau:

- Gọi p_x là pixel tại vị trí x trong ảnh I và Ψ_{px} là một mảnh với tâm là pixel p_x . Mảnh này có phần đã biết là Ψ_{px}^S và phần chưa biết Ψ_{px}^U .
- Tìm kiếm mảnh vá Ψ_{pj} (tâm là pixel p_j) giống nhất với Ψ_{px} . Pixel trung tâm p_j có các pixel xung quanh giống với phần đã biết của p_x .
- Sau đó, sao chép giá trị pixel tại p_j vào vị trí p_x .

Nhận xét

Thuật toán khôi phục từng pixel này gặp vấn đề khi chi phí tính toán cao, ngay cả khi độ phức tạp của nó có thể được giảm bớt bằng cách tìm kiếm các mảnh vá tương ứng tốt nhất trong số các ứng viên của các pixel gần nó đã được inpainted. Một hạn chế khác là khó khăn trong việc tổng hợp các texture không phẳng và điền vào những lỗ hỏng lớn và rải rác. Hơn nữa, mặc dù kết quả tốt hơn so với các phương pháp truyền thống trên các khu vực có texture, các kỹ thuật tổng hợp dựa trên pixel thường gặp vấn đề về việc lan truyền làm cho phần được phục hồi nhìn không tự nhiên.

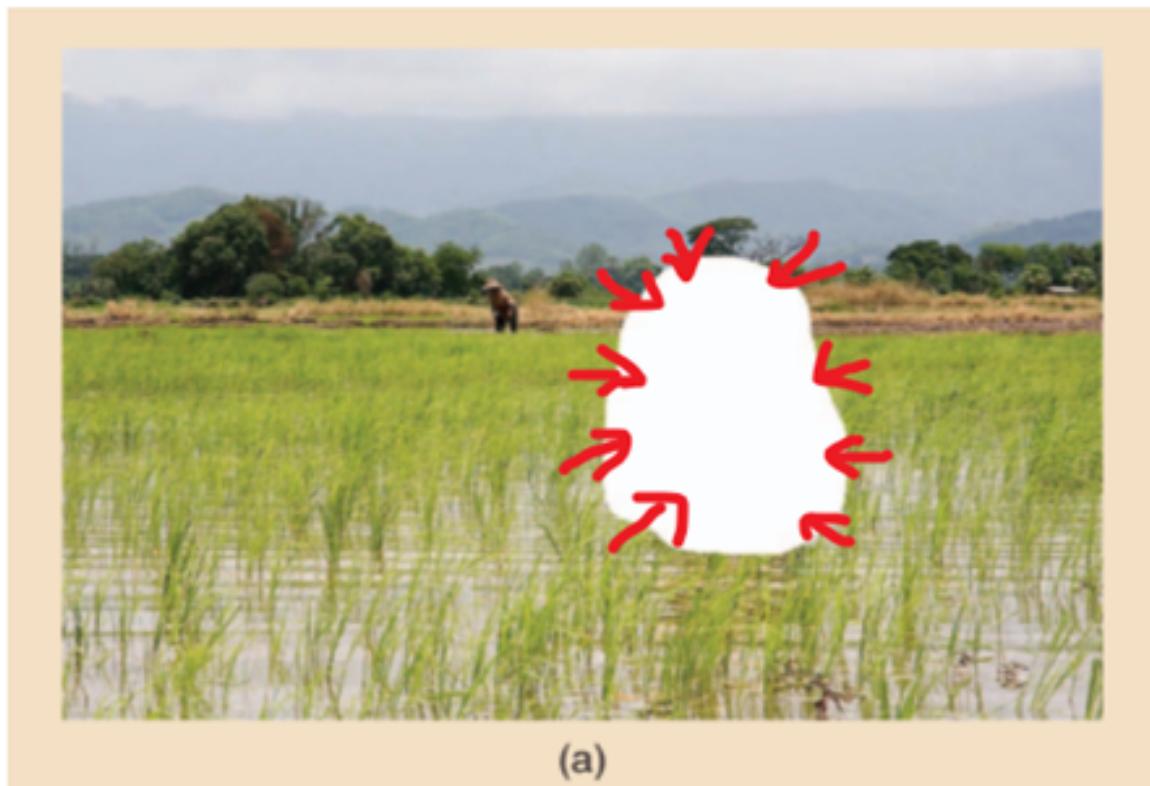


Hình II.1: Phương pháp tổng hợp texture dựa trên Pixel

1.2 Giới thiệu về phương pháp Diffusion-Based

Nguyên lý

Sử dụng giá trị các pixel từ các vùng đã biết, để truyền giá trị vào bên trong vùng chưa biết. Như nghĩa của từ diffusion [10] [11], tức khuếch tán, là sự lan truyền (truyền tải) nội dung hình ảnh từ các vùng đã biết (biên bên ngoài), và các vùng bị thiếu.



Hình II.2: Minh họa phương pháp Diffusion Based

Một ví dụ đơn giản ở đây là chính là phương pháp trung bình, khi ta sẽ trung bình cộng các pixel đã biết xung quanh (giống phương pháp làm trơn trung bình), hay gaussian, lấy trọng số các pixel đã biết xung quanh. Dựa vào nguyên tắc như trên, các pixel xung quanh có ảnh hưởng nhất định đến pixel cần điền vào.

Ví dụ phương pháp trung bình:

$$f(i, j) = \text{AVERAGE}(f(i - 1 : i + 2, j - 1 : j + 2))$$

Với i, j là ô cần điền, và chỉ tính trung bình các ô có giá trị pixel đã biết. Trong bài báo cáo lần này, về phương pháp Diffusion based, sẽ trình bày phương pháp nền tảng, được ra mắt trong bài báo năm 2000 bởi tác giả Bertalmio, từ phương pháp này, các thuật toán còn lại được phát triển lên, sẽ không trình bày trong bài này mà chỉ nói sơ qua.

Thuật toán

Nguyên lý: tương tự nguyên lý thuật toán, là sự lan truyền. Ở đây, tác giả [2] mô phỏng quá trình lan truyền bằng cách sử dụng PDE (phương trình vi phân) (phương trình vi phân ở đây được mô tả từ phương trình Naive-Stokes, phương trình mô tả dòng chất lỏng - pixel lan truyền giống như quá trình chất lỏng truyền vậy).

Gọi Ω là khu vực các pixel cần được điền vào, $\theta\Omega$ là khu vực đã biết xung quanh, khi duyệt qua từng pixel của ảnh (cần được inpainting), ta có thể sử dụng công thức sau:

$$I^{n+1}(i, j) = I^n(i, j) + \Delta t I_t^n(i, j), \forall (i, j) \in \Omega$$

Với

- n : Giá trị lần cập nhập thứ n (lần $n+1$ cập nhập bởi lần n)
- $I(i, j)$: vị trí pixel
- ΔT : tốc độ cập nhập giá trị pixel

Dễ thấy, qua phương trình trên, ta sẽ duyệt qua n lần các vị trí cần khôi phục, cùng với việc đã mô phỏng được sự lan truyền (ở đây là giá trị các hệ số xung quanh), thì mỗi lần duyệt, ảnh lại được tốt hơn so với ảnh trước, cho đến khi hết n lần cập nhập, hoặc đơn giản cho đến khi ta xét điều kiện ảnh sau, không còn được tốt hơn ảnh trước nhiều nữa (tức giá trị đã hội tụ lại một điểm).

Nhận xét

Điểm mấu chốt của thuật toán trên nằm ở phương trình cập nhập các giá trị pixel dựa vào các pixel đã biết, và dựa vào việc mô phỏng được quá trình lan truyền (là các trọng số).

Thuật toán trên được mô phỏng theo dạng đẳng hướng, tức ở các hướng khác nhau, sẽ có ảnh hưởng giống nhau (hướng trái ảnh hưởng giống hướng phải), ngoài ra còn có một dạng nữa là dị hướng, tức sự lan truyền được mô phỏng theo một hướng chủ đạo, các hướng khác ảnh hưởng ít hơn (ví dụ như phương pháp Coherence-enhancing diffusion (CED) của tác giả Joachim Weickert 1999).

Nhìn vào nguyên lý, ta dễ thấy, phương pháp diffusion có thể ngay lập tức điền vào các pixel

bị thiếu thông qua các pixel xung quanh, nhưng, thường chỉ được sử dụng cho các vùng bị mất nhỏ (mask nhỏ), nếu mất quá lớn, thì xung quanh không có quá nhiều dữ liệu để bù vào, làm cho hiệu suất kém đi.

Tương tự, các thuật toán sau, được phát triển lên, như phát triển về cách mô hình hóa (dị hướng, hay đẳng hướng), hay thay đổi cách của sự lan truyền (ở trên mô phỏng theo sự lan truyền của nước, mô hình hóa theo gaussian, hay theo phương trình vi phân linear heat equation trong vật lý, v.v...), tuy nhiên nhìn chung, phương pháp diffusion này vẫn phải khuyết điểm như trên.

2. Phương pháp Deep Learning

Các phương pháp truyền thống đã trình bày ở trên, có hiệu quả trong một số trường hợp kích thước mask nhỏ, nhưng lại có hiệu suất khá thấp khi kích thước mask lớn lên, và cũng như tạo ra các ảnh với không nhiều ngữ nghĩa sâu mà chỉ thường là một màu, điều này có thể giải thích vì phương pháp truyền thống (exemplar hay diffusion) thường lấy các giá trị xung quanh bù vào mà mất quá nhiều thì các giá trị xung quanh không còn đa dạng nữa.

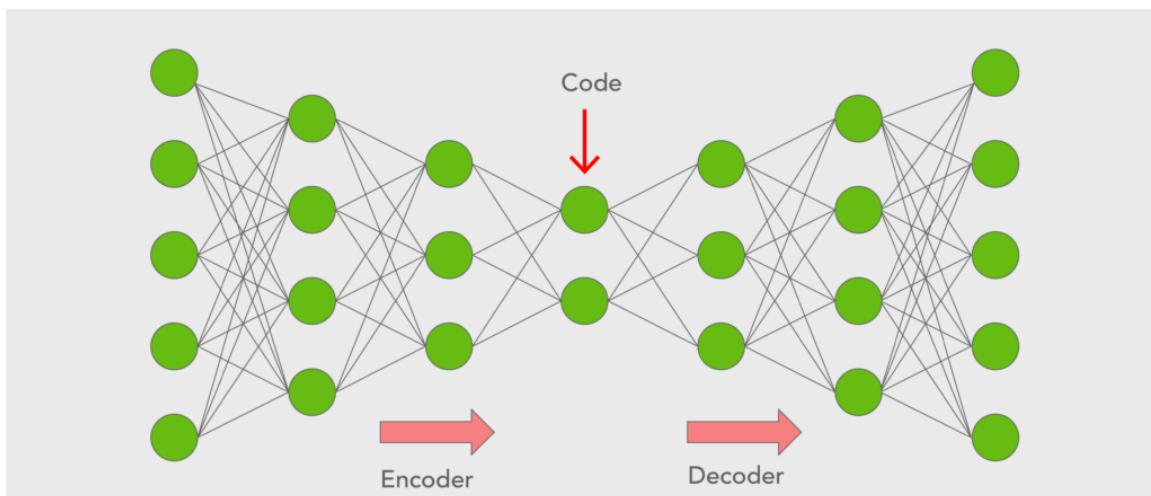
Từ đó, để giải quyết các vấn đề mà phương pháp truyền thống chưa làm được, học sâu (deep learning) [7] [13] dựa vào sức mạnh tính toán máy tính đã được tìm thấy là mô hình có thể giải quyết vấn đề phục hồi ảnh này (image inpaiting), một cách tốt hơn.

Về cơ bản, học sâu là một nhánh học khác của học máy, lấy cảm hứng từ cấu trúc và chức năng của bộ não con người. Nó tận dụng các mạng lưới thần kinh nhân tạo để tìm hiểu các mẫu và cách biểu diễn phức tạp từ dữ liệu, cho phép hệ thống đưa ra các quyết định và dự đoán thông minh.

Bài làm sẽ giới thiệu các mô hình có thể ứng dụng vào trong vấn đề hồi phục ảnh này, sau đó tiến tới chọn một phương pháp để trình bày.

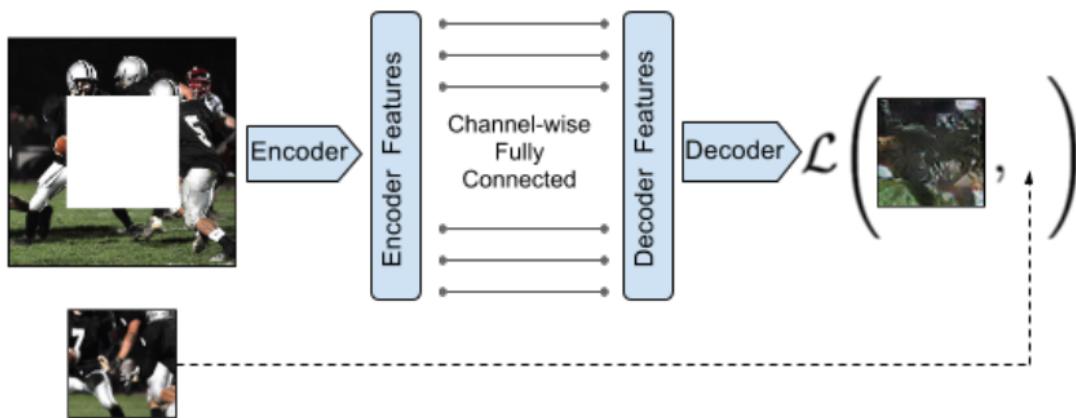
2.1 Encoder Decoder

Ta sẽ xem xét quá trình train của mô hình, để trả lời được câu hỏi làm sao mô hình tìm được pixel để điền vào trong vùng bị mất.



Hình II.3: Mô hình encoder - decoder

Với dữ liệu đầu vào sẽ là các tập ảnh cần được điền vào, ảnh xác thực sẽ là ảnh nguyên gốc, như sau (nói sơ qua, sau đó qua phần phương pháp sẽ trình bày rõ hơn).



Hình II.4: Ví dụ Encoder - Coder training [9]

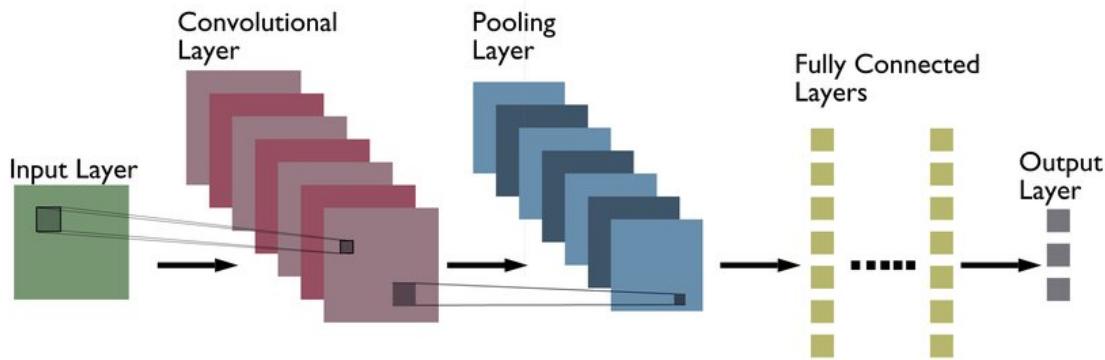
Thì mỗi pixel của ảnh hư, sẽ được xem như một neural trong mô hình deep learning này, sau đó sẽ quá trình encoder (giảm số neural xuống), sau đó decoder (tăng số neural lên lại), sao cho tại level cuối, ta có số neural tương ứng với số pixel ảnh mới, tức ảnh đã được phục hồi.

Điều này giống như, khi giảm số neural xuống, ta muốn mô hình cố gắng học các pixel tốt để giữ lại ý nghĩa, trích xuất đặc trưng, sau đó dùng những đặc trưng tốt này để tái tạo lại ảnh mới (ảnh được phục hồi).

Sau khi phục hồi như vậy thì ta so với ảnh xác thực, có loss function, loss function ở đây sẽ là thể hiện cái mức độ tốt của ảnh điền (không nhất thiết phải là trùng với ảnh gốc - vì thường không đủ thông tin - sẽ được trình bày rõ hơn ở phần phương pháp), và ta cố gắng tối thiểu hàm loss này, để mô hình học được mối liên hệ/đặc trưng, mà điền vào.

2.2 CNN - Convolutional Neural Network

Khác với encoder-decoder như trên, CNN, sẽ biến đổi dữ liệu bằng các filters (như trong biên cạnh có sobel, laplace...), để khiến dữ liệu trở nên tốt hơn, sau đó đưa qua train tiếp như neural network bình thường (mạng Fully connected layer - để có thêm nhiều sự liên kết giữa các thông tin).

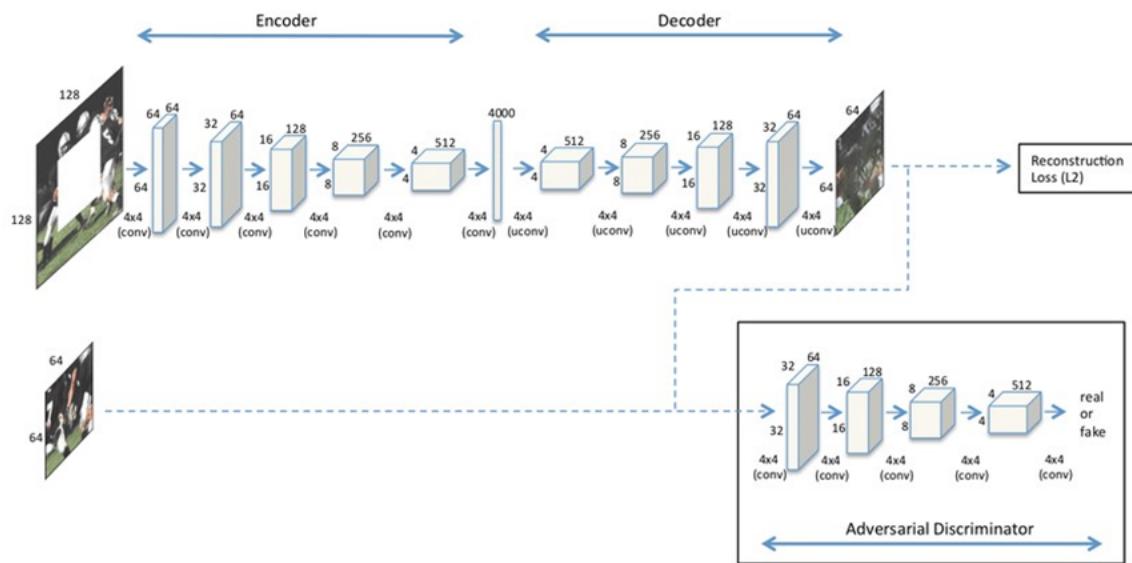


Hình II.5: Minh họa mô hình CNN.

Như hình trên, quá trình trích xuất tính năng sẽ thực hiện qua các lớp gồm:

- Convolutional Layer: Trích xuất các thông tin khác nhau từ input đầu vào. (Tức ví dụ ở đây sẽ là các filter 3x3, ban đầu được tạo ngẫu nhiên, sau đó mô hình sẽ học để thay đổi giá trị trong các filter này).
- Sau đó, Pooling Layer: nhằm giảm kích thước của dữ liệu sau khi trích xuất thông tin, ví dụ như xét phạm 2x2, ta lấy MaxPooling (giá trị tối đa trong 2x2), để giảm chiều dữ liệu xuống → rút trích đặc trưng.

Ví dụ ứng dụng vào iamge inpainting



Hình II.6: Minh họa mô hình CNN trong image inpainting.

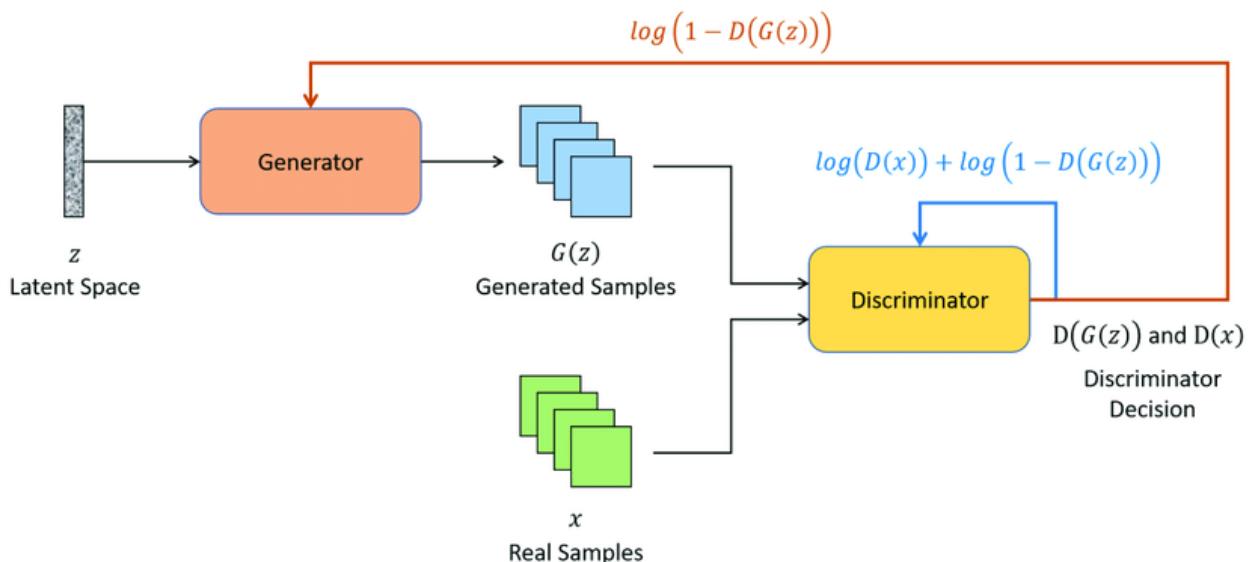
Sau khi được làm tốt dữ liệu qua các lớp filters, thì đây được xem là lớp input mới, sẽ được đưa vào Lớp Fully Connected Layers (FCN - là neural network với nhiều hidden layer

được sắp xếp có mục đích), rồi đến output layer, đến này quá trình train diễn ra như train quá trình neural network cơ bản (train từ input mới \rightarrow FCNs \rightarrow Output). (FCN chính là neural network nhắc ở trên, vì các filter chỉ lấy khoảng 3x3, nên FCN dễ liên kết nhiều thông tin lại với nhau).

Nhận xét: Sự khác biệt ở đây với Encoder - Decoder, khi mà Encoder - Decoder, chỉ train thuần, tận dụng sức mạnh máy tính để tìm mối liên hệ giữa các pixel, thì CNN sẽ biến đổi dữ liệu qua các filters, nhằm mục đích biểu diễn tốt hơn trước, sau đó mới dùng FCN.

2.3 GANs - Generative Adversarial Networks

Trước tiên ta có thể xem qua về mô hình GANs [5]:



Hình II.7: Minh họa mạng GANs đơn giản.

Như hình trên GANs bao gồm hai thành phần chính đó là:

- Generator: tạo ảnh, với mục tiêu là từ nhiễu/ dữ liệu bất kì, cố gắng tạo được ảnh/output giống như ảnh/output thực nhất có thể. (Có thể hiểu là cố gắng đánh lừa bộ phận phân biệt Discriminator)
- Discriminator: Bộ phân phân biệt, nhận vào là tập ảnh giả tạo được lớp Generator, và tập ảnh thật từ bộ dữ liệu. Lớp này cố gắng sao phân biệt đúng đâu là ảnh thật, đâu là ảnh giả từ generator.

Vì sao cần GANS? Vì bài toán của ta là điền giá trị pixel vào, mà các mô hình trước (Encoder-Decoder, CNN,...) thường bị fix cứng giá trị loss function, thành ra ảnh thường sẽ không được tự nhiên, và thường sẽ khá là mờ (vì thường là lấy pixel xung quanh), nên người ta mới thêm một lớp Discriminator phân loại, nhằm giúp mô hình tạo ảnh một cách tự nhiên hơn.

Tất nhiên, hai lớp Generator và Discriminator ở đây đơn giản là hai lớp neural network (với các bộ hệ số ứng với các lớp hidden network), một bên là tạo, một bên phân loại. Và sẽ cập nhập hệ số dựa trên minimax game, Với Loss Function thể hiện mức độ Discriminator dự đoán đúng, thì Generator muốn giảm thiểu hệ số này xuống (đánh lừa Discriminator). Về Loss Function, công thức:

$$\min_G \max_D \mathbb{E}_{x \in \mathcal{X}} [\log(D(x))] + \mathbb{E}_{z \in \mathcal{Z}} [\log(1 - D(G(z)))]$$

Hình II.8: Minh họa một loại loss function cho mô hình GANs.

Công thức trên đơn giản là bắt nguồn từ công thức Binary Cross Entropy, với:

$$Loss = y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

Với

- y : là output xác thực
- \hat{y} : là output mà mô hình dự đoán

Là loss function khi so sánh một lần ($\hat{y} = 0$, hoặc 1), và $D(x)$ chính là mô hình dự đoán của ta, có hai trường hợp như sau:

- $y = 0$: $Loss = \log(1 - D(G(z)))$
- $y = 1$: $Loss = \log(D(x))$

Mà vì mỗi lần ta đưa vào nhiều ảnh cho discriminator phân loại, nên ta sẽ lấy trung bình nhiều lần này lại, tạo thành công thức Loss của mô hình như ảnh II.7.

2.4 Nhận xét

Nhận xét: 3 mô hình trên là 3 mô hình chính của deep learning sử dụng cho image inpainting, và người ta vẫn có thể áp dụng các mô hình lại với nhau để giải bài toán, mỗi mô hình từng công đoạn như Contextual Attention [14] (1 CNN cho việc điền vào, 1 GAN cho việc cải tiến kết quả), hay Edge Connect [8] (1 GAN cho việc training phát hiện biên cạnh, 1 GAN cho việc điền vào).

Và ta có thể nhận thấy rằng, sự phát triển đi lên của phương pháp, cũng đồng thời làm tăng kiến trúc mạng, độ phức tạp, nhưng cái tăng quan trọng nhất chính là thông tin (từ Encoder-Decoder chỉ sử dụng sức mạnh tính toán, CNN có filter, GANs thêm lớp discriminator, hay

kết hợp hai lớp CNN-GANs, GANs-GANs), kiến trúc mạng càng tăng là để tăng lượng thông tin để điền vào.

Và từ đó, nhóm có tìm hiểu thêm hai phương pháp có cách tìm kiếm thông tin khác, đó là dùng Fourier để biểu diễn thông tin (miền tần số) cho mạng, là Lama, CMGAN [12], [15].

Để dễ hình dung, ta có thể so sánh vài phương pháp và đưa ra nhận xét như sau - ở đây sẽ so sánh 3 phương pháp là Lama, CMGAN (biểu diễn thông tin trong miền tần số), và dùng thêm Edge Connect (state of the arts).

Phương pháp	Input-Training	Cách lấy thông tin chính	Ứng dụng
Edge Connect	GANs1: ảnh hư + mask + biên cạnh hư GANs2: Ảnh hư + GANs1 (Thông tin biên cạnh)	1 GANs edge detection 1GANs inpainting	Hồi phục và giữ thông tin biên cạnh
Lama	Ảnh hư + Mask	Fourier	Loại bỏ đối tượng (Video demo)
CM-GAN	Ảnh hư + Mask	Fourier + GANs	Hồi phục, với độ chính xác cao hơn

Bảng II.1: So sánh các phương pháp Deep Learning.

Ta có một số nhận xét sau:

- Input training thì Edge Connect có nhiều hơn là rõ, vì cần thêm thông tin về biên cạnh.
- Về cách lấy thông tin chính, thì Edge Connect dùng 2 mạng GANs, thành ra độ phức tạp lớn hơn nhiều, trong khi Lama chính yếu là dùng Fourier biểu diễn thông tin (tất nhiên có nhiều version, còn GANs thì Fourier, và dùng thêm kiến trúc GANs cascade model (phân tầng), thường được dùng trong lĩnh vực đòi hỏi sự chính xác cao, ý tế, v...)

Vì vậy, CM-GAN, nhóm có nhận xét là phức tạp hơn và dư với nhu cầu ứng dụng loại bỏ đối tượng, và Edge Connect thì phức tạp, thông tin biên cạnh đôi khi cũng không cần thiết lắm, ta có thể điền vào hợp lý là được, còn Lama thì khá phù hợp với ứng dụng này, nên nhóm sẽ chọn Lama cho phần trình bày phương pháp.

Chapter III

Phương pháp Lama

1. Dữ liệu huấn luyện

Để huấn luyện mô hình ta cần dữ liệu đầu vào bao gồm hai phần mặt nạ (mask) và ảnh đầu vào còn nguyên vẹn. Sau đó ảnh sẽ được gắn mặt nạ vào để tạo ra một ảnh bị mất mát thông tin ở phần mặt nạ. Và dữ liệu xác thực cũng chính là ảnh gốc khi chưa bị gắn mặt nạ vào. Do đó khi chuẩn bị dữ liệu ta cần chuẩn bị 2 phần bao gồm mặt nạ và ảnh gốc.

1.1 Thuật toán Mask Generation

Ở đây nhóm tác giả không tự tạo mặt nạ bằng các phương pháp thủ công mà sử dụng các thuật toán để tạo ra các mặt nạ một cách tự động có thể điều chỉnh theo tham số bao gồm một thuật toán tạo mặt nạ ngẫu nhiên và một thuật toán tạo mặt nạ dựa trên phân đoạn (segmentation-based).

Thuật toán Random Mask Generation

Thuật toán này giúp tạo ra mặt nạ một cách ngẫu nhiên, nhóm tác giả đã lựa chọn một thuật toán tạo mặt nạ lớn. Mỗi mặt nạ sẽ là sự kết hợp giữa một chuỗi các đa giác (wide mask) hoặc các hình chữ nhật có tỉ lệ kích thước ngẫu nhiên (box mask), hai loại mặt nạ này sẽ được tạo ra một cách ngẫu nhiên theo phân phối chuẩn có thể điều chỉnh theo tham số p_{irr} được truyền vào. Nếu tỉ lệ ngẫu nhiên được tạo ra nhỏ hơn p_{irr} thì thuật toán sẽ tạo wide mask còn lớn hơn thì thuật toán sẽ tạo ra box mask.

```
import numpy as np

def gen_large_mask(img_h, img_w, n):
    """ img_h: int, an image height
        img_w: int, an image width
        marg: int, a margin for a box starting coordinate
        p_irr: float, 0 <= p_irr <= 1, a probability of a polygonal chain mask

        min_n_irr: int, min number of segments
        max_n_irr: int, max number of segments
        max_l_irr: max length of a segment in polygonal chain
        max_w_irr: max width of a segment in polygonal chain
    """
    pass
```

```

min_n_box: int, min bound for the number of box primitives
min_n_box: int, max bound for the number of box primitives
min_s_box: int, min length of a box side
max_s_box: int, max length of a box side"""

mask = np.ones((img_h, img_w))

if np.random.uniform(0, 1) < p_irr: # generate polygonal chain
    n = np.random.uniform(min_n_irr, max_n_irr) # sample number of segments

    for _ in range(int(n)):
        y = np.random.uniform(0, img_h) # sample a starting point
        x = np.random.uniform(0, img_w)

        a = np.random.uniform(0, 360) # sample angle
        l = np.random.uniform(10, max_l_irr) # sample segment length
        w = np.random.uniform(5, max_w_irr) # sample a segment width

        # draw segment starting from (x, y) to (x_, y_) using brush of width w
        x_ = x + l * np.sin(a)
        y_ = y + l * np.cos(a)

        # Assuming gen_segment_mask is defined somewhere
        gen_segment_mask(mask, start=(x, y), end=(x_, y_), brush_width=w)
        x, y = x_, y_

else: # generate Box masks

    n = np.random.uniform(min_n_box, max_n_box) # sample number of rectangles

    for _ in range(int(n)):
        h = np.random.uniform(min_s_box, max_s_box) # sample box shape
        w = np.random.uniform(min_s_box, max_s_box)

        x_0 = np.random.uniform(marg, img_w - marg + w) # sample upper-left coordinates of box
        y_0 = np.random.uniform(marg, img_h - marg - h)

        # Assuming gen_box_mask is defined somewhere
        gen_box_mask(mask, size=(img_w, img_h), masked=(x_0, y_0, w, h))

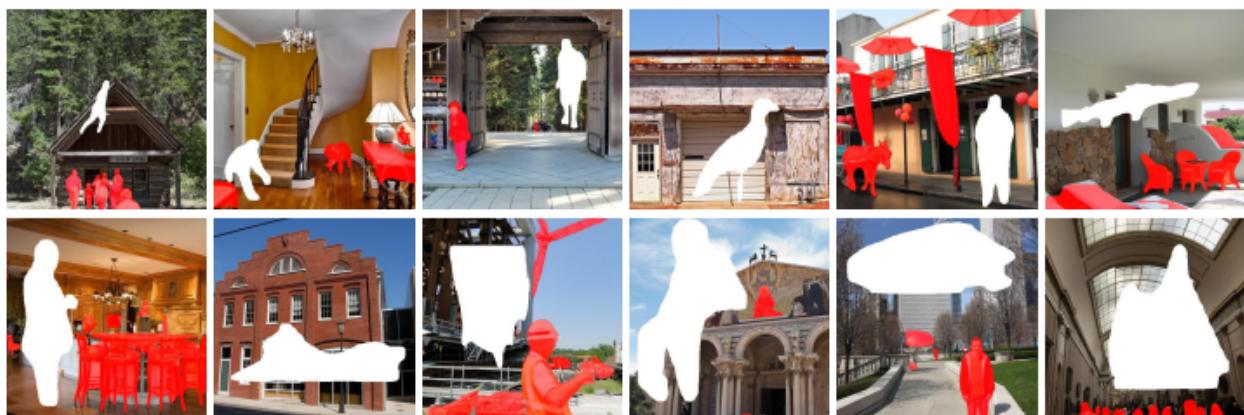
return mask

```

Thuật toán Segmentation-based Mask

Bên cạnh các mặt nạ được tạo ngẫu nhiên, nhóm tác giả cũng sử dụng các mặt nạ dựa trên phân đoạn, để đảm bảo rằng các kết luận thu được từ các mặt nạ không đều ngẫu nhiên cũng có hiệu lực đối với các đối tượng, hình dạng và kích thước thực tế trong thế giới thực. Bộ mặt nạ Segm được thiết kế để mô phỏng ứng dụng thực tế của việc loại bỏ đối tượng, ví dụ như trong một trình chỉnh sửa ảnh.

Các mặt nạ Segm được tạo ra sử dụng segmentation-based mask generator. Công cụ tạo sinh mặt nạ này sẽ trích xuất bóng của các đối tượng có trong ảnh sử dụng công cụ Detectron2, sau đó chúng sẽ tạo ra mặt nạ từ bóng của các đối tượng vừa trích xuất được. Tập ảnh được lấy để trích xuất bóng được lấy từ bộ dữ liệu Places.



Hình III.1: Phần màu đỏ: các đối tượng trích xuất được từ công cụ Detectron2, Phần màu trắng: Segm mask lấy được bằng việc trích xuất bóng của đối tượng từ ảnh khác

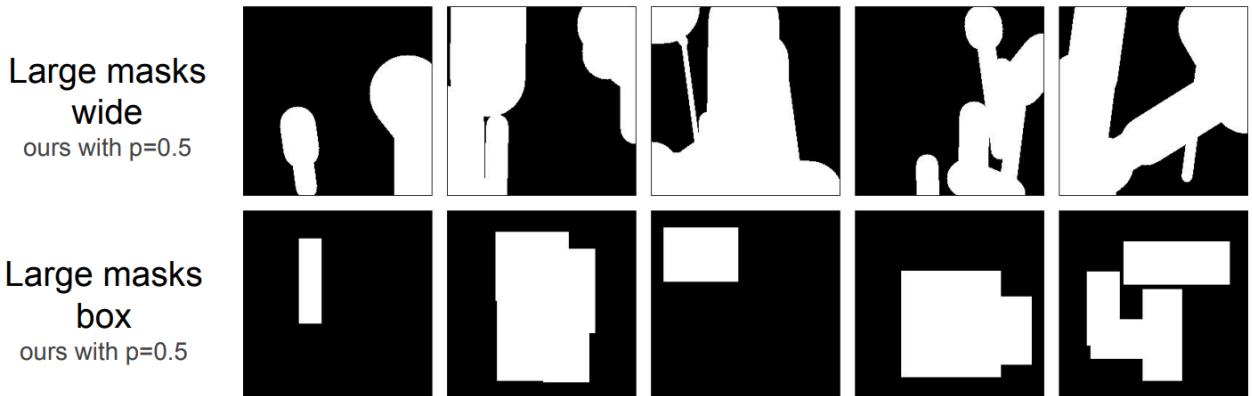
1.2 Places Dataset

Bộ dữ liệu **Places365** là một bộ dữ liệu nhận diện cảnh quan. Nó bao gồm 10 triệu hình ảnh, trong đó có 434 lớp cảnh khác nhau. Có hai phiên bản của bộ dữ liệu: Places365-Standard với 1.8 triệu hình ảnh cho tập huấn luyện và 36000 hình ảnh xác nhận từ K=365 lớp cảnh, và Places365-Challenge-2016, trong đó kích thước tập huấn luyện tăng lên đến 6.2 triệu hình ảnh bổ sung, bao gồm 69 lớp cảnh mới (đến tổng cộng 8 triệu hình ảnh huấn luyện từ 434 lớp cảnh). Trong phương pháp Lama tác giả sử dụng phiên bản Places365-Standard để huấn luyện mô hình. Cụ thể như sau:

- **Tập huấn luyện (Training)** Sử dụng tất cả 1.8 triệu ảnh cho tập huấn luyện, mỗi ảnh có kích thước 512×512 pixel.
- **Tập đánh giá (Validation)** Để thực hiện đánh giá trong quá trình huấn luyện, để theo dõi hiện tượng overfitting và chọn điểm checkpoint tốt nhất, tác giả đã chuẩn bị một tập đánh giá bao gồm 2000 cặp ảnh-mặt nạ. Các ảnh trong tập đánh giá được lấy ngẫu nhiên

từ tập dữ liệu đánh giá của bộ dữ liệu Places. Các mặt nạ cho tập đánh giá được chuẩn bị bằng thuật toán tạo mặt nạ dựa trên phân đoạn (segmentation-based mask generation algorithm).

- Tập thử nghiệm (Test)** Để thực hiện đánh giá cuối cùng, tác giả đã chuẩn bị bốn tập kiểm tra - ba tập với mặt nạ ngẫu nhiên với độ rộng khác nhau (hẹp, trung bình, rộng) và một tập với mặt nạ dựa trên phân đoạn. Các tập kiểm tra với mặt nạ ngẫu nhiên chứa 30000 cặp ảnh-mặt nạ và tập chứa mặt nạ dựa trên phân đoạn chứa 4000 cặp. Tất cả các ảnh được lấy ngẫu nhiên từ tập kiểm tra của bộ dữ liệu Places.



Hình III.2: Ví dụ của Wide Masks và Box Masks với $p_{irr} = 0.5$

Thiết lập các tham số cho mặt nạ

	p_irr	Irregular Masks				Box-shaped masks					
		min_n_irr	max_n_irr	max_l_irr	max_w_irr	min_n_box	max_n_box	min_s_box	max_s_box	marg	
256	Narrow*	1	4	50	40	10	-	-	-	-	-
	Medium	0.77	4	5	100	50	1	5	10	50	0
	Wide	0.77	1	5	200	100	1	3	30	150	10
	Train	0.5	1	5	200	100	1	4	30	150	10
512	Narrow	1	4	70	100	20	-	-	-	-	-
	Medium	0.77	4	10	200	100	1	5	30	150	0
	Wide	0.77	1	5	450	250	1	4	30	300	10

Hình III.3: Tham số của mặt nạ được tạo với các độ rộng: hẹp (narrow), trung bình (Medium) và rộng (Wide)

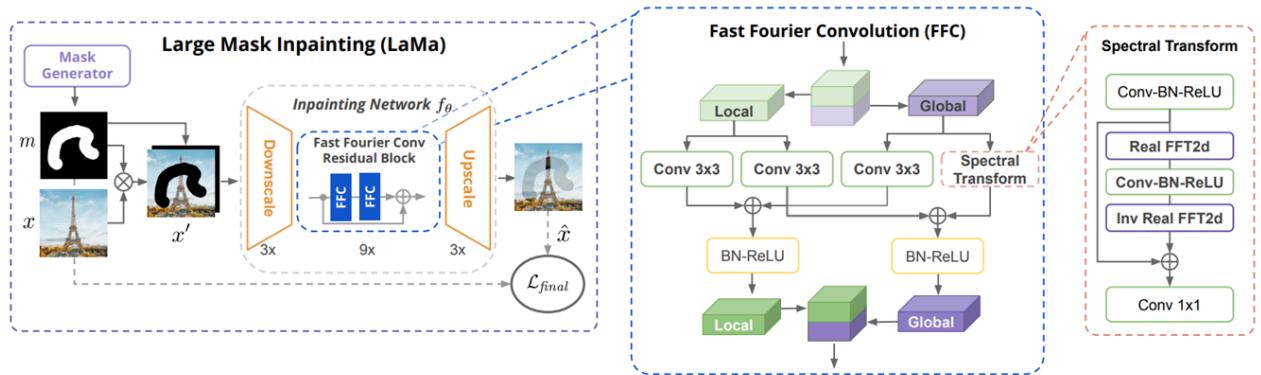
2. Kiến trúc mạng

2.1 Đặt vấn đề

Trong những trường hợp khó khăn, ví dụ như phần bị hỏng quá lớn, quá trình inpainting đòi hỏi phải xem xét ngữ cảnh toàn cục vì lúc đó các thông tin cục bộ không còn nhiều giá trị do mất mát quá lớn. Vì vậy, một kiến trúc tốt nên có các trường tiếp nhận các thông tin rộng nhất có thể ngay từ lúc bắt đầu trong quá trình xử lý. Các mô hình convolutional thông thường, ví dụ như ResNet, gấp vấn đề với việc trường tiếp nhận thông tin phát triển quá chậm. Dẫn đến việc các đặc trưng thu thập được có thể không đủ, đặc biệt là ở các tầng đầu của mạng,

do các kernel convolutional được sử dụng thường nhỏ (ví dụ: 3×3). Do đó, nhiều tầng trong mạng sẽ thiếu cảnh toàn cầu và sẽ lãng phí chi phí tính toán và tham số để tạo ra nó. Đối với các mặt nạ rộng, toàn bộ kernel của một trường nhận thức ở vị trí cụ thể có thể nằm hoàn toàn bên trong mặt nạ, do đó chỉ xem các điểm ảnh còn thiếu. Và những điểm ảnh đó thì hoàn toàn không có giá trị trong việc trích xuất đặc trưng.

2.2 Tổng quát



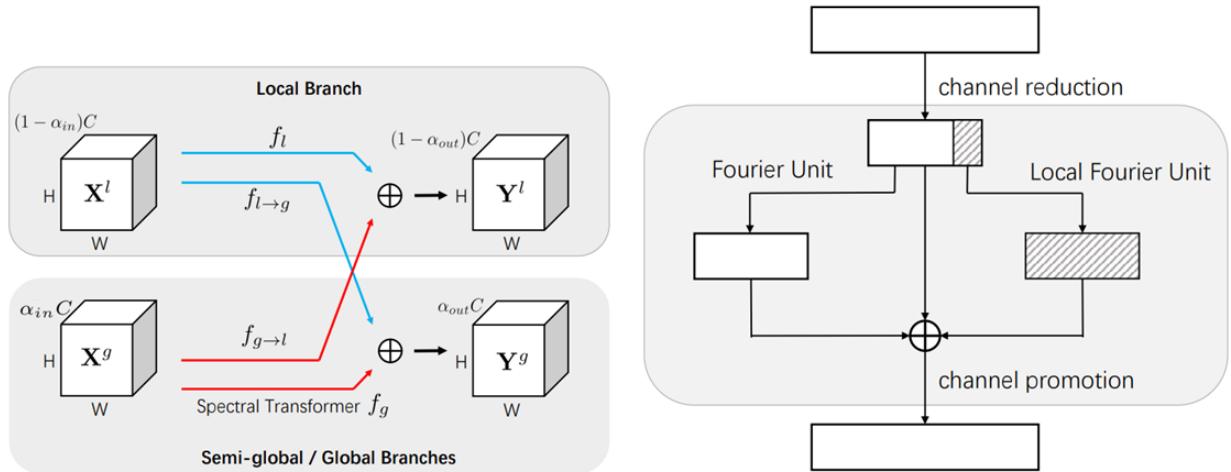
Hình III.4: Kiến trúc mạng Lama sử dụng FFC

Mục tiêu của phương pháp là tìm cách điền vào một hình ảnh màu x được che phủ bởi một mặt nạ nhị phân của tập các điểm ảnh mà ta không biết m , hình ảnh bị che phủ bởi mặt nạ được ký hiệu là $x \odot m$. Mặt nạ m được xếp chồng với hình ảnh bị che phủ $x \odot m$, tạo ra một tensor đầu vào có bốn kênh $x' = \text{stack}(x \odot m, m)$. Sử dụng một mạng inpainting feed-forward $f_\theta(\cdot)$, được xem là mạng tạo sinh (generator). Bằng cách sử dụng x' , mạng sẽ dùng các lớp convolutional để tạo ra ảnh kết quả đã được inpainted với ba kênh màu $\hat{x} = f_\theta(x')$. Việc huấn luyện được thực hiện trên một tập dữ liệu gồm các cặp (hình ảnh, mặt nạ) thu được từ hình ảnh thực và mặt nạ được tạo ra ngẫu nhiên đã được đề cập ở phần trên.

2.3 Fast Fourier Convolution

Fast Fourier convolution (FFC) [3] là một phương pháp cho phép mô hình có thể sử dụng ngữ cảnh toàn cục ở các tầng đầu của mạng. FFC dựa trên phép biến đổi Fourier nhanh (FFT) [1] theo từng kênh và có trường nhận thức bao phủ toàn bộ hình ảnh. FFC chia các kênh thành hai nhánh song song:

- Nhánh cục bộ sử dụng convolution thông thường
- Nhánh toàn cầu sử dụng real FFT để thu thập đặc trưng toàn cục.



Hình III.5: Fast Fourier Convolution

Với $X \in \mathbf{R}^{H \times W \times C}$ là đặc trưng đầu vào của FFC, với $H \times W$ là kích thước dài và rộng trong miền không gian và C đại diện cho số kênh. Ở đầu vào của FFC, X sẽ được tách dựa trên số kênh, $X = \{X^l, X^g\}$. Nhánh cục bộ $X^l \in \mathbf{R}^{H \times W \times (1-\alpha_{in})C}$ được sử dụng để học các đặc trưng từ các phần lân cận và nhánh toàn cục $X^g \in \mathbf{R}^{H \times W \times \alpha_{in}C}$ được sử dụng để học được ngữ cảnh với miền thu nhận rộng. Với $\alpha_{in} \in [0, 1]$ là tỉ lệ của số lượng kênh sẽ được đưa vào nhánh toàn cục.

Để dễ hiểu, cho rằng output có kích thước bằng với input. Gọi $Y \in \mathbf{R}^{H \times W \times C}$ là kết quả output. Tương tự như X , $Y = \{Y^l, Y^g\}$. Công thức dưới đây mô tả việc cập nhật của mỗi khối FFC .

$$\begin{aligned} Y^l &= Y^{l \rightarrow l} + Y^{g \rightarrow l} = f_l(X^l) + f_{g \rightarrow l}(X^g), \\ Y^g &= Y^{g \rightarrow g} + Y^{l \rightarrow g} = f_g(X^g) + f_{l \rightarrow g}(X^l). \end{aligned}$$

Hình III.6: Công thức mô tả đầu vào và đầu ra của khối FFC

Đối với thành phần $Y_{l \rightarrow l}$ với mục đích thu nhận các thông tin cục bộ, ta sử dụng các tích chập thông thường. Tương tự, hai thành phần ($Y_{g \rightarrow l}$ và $Y_{l \rightarrow g}$) cũng sử dụng các tích chập thông thường để thu thập thông tin và trao đổi giữa hai nhánh. Phần đặc biệt của kiến trúc đến từ thành phần $Y_{g \rightarrow g}$ nơi mà ta có thể trích xuất đặc trưng toàn cục trên miền tần số từ những lớp đầu tiên của mạng. Thành phần này gồm hai đơn vị nhỏ:

- Đơn vị là Fourier Unit (FU) sử dụng Fast Fourier Transform (FFT) để chuyển dữ liệu từ miền không gian sang miền tần số thu thập đặc trưng và chuyển dữ liệu ngược về lại miền không gian.
- Đơn vị Local Fourier Unit (LFU) cũng tương tự với đơn vị Fourier Unit nhưng chỉ tiến hành thu thập đặc trưng trên một phần tư số kênh đầu vào trong khi FU được dùng cho toàn bộ đặc trưng đầu vào. Đơn vị này nhằm mục tiêu thu thập được các thông tin bán toàn cục trong miền tần số.

Do đó thành phần này vừa trích xuất được các đặc trưng rộng trong miền tần số nhưng vẫn không quên chú ý đến những phần đặc trưng bán toàn cục. Để giúp mô hình có thể vừa hiểu được các kết cấu chung của ảnh cần phục hồi nhưng vẫn có thể tập trung vào những phần nổi bật của ảnh giúp cải thiện sự sắc nét và tự nhiên của ảnh kết quả sau khi phục hồi.

3. Loss function

3.1 Perceptual Loss

Ta có nhận xét rằng bài toán image inpainting này là bài toán mơ hồ, vì có thể có nhiều đáp án điền vào chỗ trống bức ảnh. Vì vậy, loss function so sánh sự khác biệt với ảnh xác thực sẽ không khả thi, vì:

- Không đủ thông tin điền vào.
- Mục đích là để điền vào một cách "phù hợp".

Tác giả bài báo Lama đã đề xuất sử dụng Perceptual loss, khi mà so sánh sự khác biệt giữa hai feature map từ hai ảnh được tái tạo, ảnh xác thực này. Từ hai ảnh trên, ta sẽ đưa qua pretrain model neural network, được hai feature map, và so sánh trên này:

$$\mathcal{L}_{HRFPL}(x, \hat{x}) = \mathcal{M}([\phi_{HRF}(x) - \phi_{HRF}(\hat{x})]^2),$$

Hình III.7: Perceptual Loss

Với,

- Ảnh đi qua pretrained model ϕ
- Và sau đó so sánh khoảng cách.

3.2 Adversarial loss

Ngoài Perceptual Loss như trên, nhóm tác giả còn định nghĩa thêm mạng discriminator $D_\varepsilon(\cdot)$ (GAN), để tăng chất lượng tạo ảnh.

Với hàm loss function của discriminator:

$$\begin{aligned}\mathcal{L}_D = & -\mathbb{E}_x \left[\log D_\xi(x) \right] - \mathbb{E}_{x,m} \left[\log D_\xi(\hat{x}) \odot m \right] \\ & - \mathbb{E}_{x,m} \left[\log (1 - D_\xi(\hat{x})) \odot (1 - m) \right]\end{aligned}\quad (2)$$

$$\mathcal{L}_G = -\mathbb{E}_{x,m} \left[\log D_\xi(\hat{x}) \right] \quad (3)$$

$$L_{Adv} = sg_\theta(\mathcal{L}_D) + sg_\xi(\mathcal{L}_G) \rightarrow \min_{\theta, \xi} \quad (4)$$

Hình III.8: Adversarial Loss 1

Với:

- m: mask.
- x: ảnh xác thực.
- \hat{x} : ảnh được hồi phục.

Ở đây ngoài hai công thức giống trong mạng GANs, $E_x[\log D_\varepsilon(x)]$ và $E_{x,m}[\log D_\varepsilon(\hat{x})]$, tác giả còn sử dụng khi nhân thêm từng phần vùng mất mát $E_{x,m}[\log D_\varepsilon(\hat{x})] \bullet m$ và vùng không mất mát: $E_{x,m}[\log(1 - D_\varepsilon(\hat{x}))] \bullet (1 - m)$

Ngoài ra, tác giả còn sử dụng thêm hai công thức để ổn định cho quá trình training:

- Gradient penalty: $R1 = \| \nabla D_\varepsilon(\hat{x}) \|^2$
- Áp dụng thêm perceptual loss trong Discriminator.

Lấy trọng số các loss function trên, có được công thức loss function cuối cùng như sau:

$$\mathcal{L}_{final} = \kappa L_{Adv} + \alpha \mathcal{L}_{HRFPL} + \beta \mathcal{L}_{DiscPL} + \gamma R_1$$

Hình III.9: Final Loss function

Trong đó, tác giả nhận xét:

- $L_{adv} v L_{DiscPL}$ thường sẽ giúp mô hình tạo một cách tự nhiên hơn, đồng thời chú ý thông tin cục bộ.

- Và L_{HRFPL} thường giúp mô hình giữ vững tính chất toàn cục của ảnh.

Chapter IV

Kết quả thử nghiệm

1. Phần code

Code được sử dụng sẽ là mô hình pretrained Lama được lấy từ github của tác giả, và sử dụng trên google colab, ta cần chỉnh một vài dòng như sau:

Vì mô hình pretrained cũ theo bài tác giả đã bị xóa, ta thay bằng link khác như sau:

```
# !curl -L $(yadisk-direct https://disk.yandex.ru/d/ouP6l8VJ0HpMZg) -o big-lama.zip  
!curl -LJO https://huggingface.co/smartywu/big-lama/resolve/main/big-lama.zip  
!unzip big-lama.zip
```

Trong file *edit/content/lama/saicinpainting/training/modules/fake_fakes.py*

```
#from kornia import SamplePadding  
from kornia.constants import SamplePadding
```

Trong file *edit/content/lama/bin/predict.py* (hàng 44)

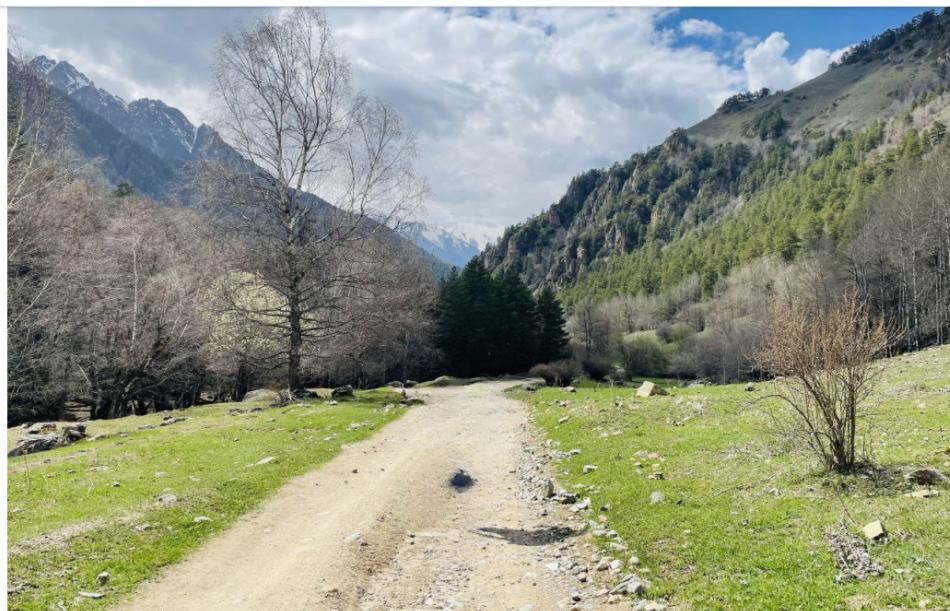
```
#device = torch.device(predict_config.device  
device = torch.device("cpu")
```

2. Kết quả

2.1 Kết quả 1



Hình IV.1: Input đầu vào 1



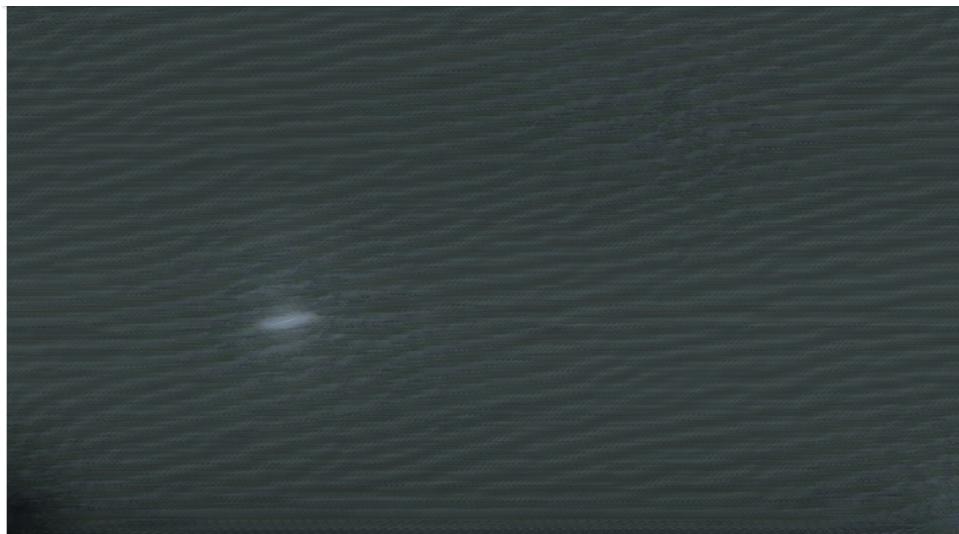
Hình IV.2: Output1

Nhận xét: Việc điền sẽ chỉ ảnh hưởng đến vùng mask mình truyền vào, nên khi mình chọn vùng con người, được thay thế, nhưng cái bóng vẫn còn giữ lại, ta có thể tiếp tục sử dụng hình này để xóa cái bóng đi tiếp.

2.2 Kết quả 2



Hình IV.3: Input đầu vào 2



Hình IV.4: Output2

Nhận xét: Ở đây em thử xem nếu tô hết tất cả ảnh thì output sẽ ra sao, và kết quả ra các màu lặp lại như gần nhau (như sóng).

Chapter V

Kết luận và hướng phát triển

1. Kết luận

Trong đồ án lần này, nhóm đã tìm hiểu chủ đề Image Inpainting (hồi phục ảnh), nguyên lý và các phương pháp của chủ đề. Đồng thời nhóm cũng đã trình bày được phát biểu bài toán và ẩn số cần tìm. Các công trình nghiên cứu liên quan gồm các giải pháp truyền thống (Exemplar, Diffusion Based), và tìm hiểu cách ứng dụng các mô hình Deep Learning ứng dụng vào (CNN, GANs,...)

Ngoài ra, nhóm đã so sánh và trình bày phương pháp Lama ứng dụng vào Image Inpainting với nguyên lý nền tảng là chuyển sang vùng không gian (sử dụng Fourier) để xử lý.

Cuối cùng, nhóm chọn demo bằng ứng dụng **loại bỏ đối tượng** sử dụng Lama làm một phần của bài tìm hiểu.

2. Hướng phát triển

Từ bài báo cáo về image inpainting này, ta có thể phát triển lên theo các hướng sau:

- Phát triển kĩ thuật để có thể ứng dụng vào video inpainting.
- Có thể phát triển cách xử lý thông tin tần số (Frequency domain) một cách tốt hơn.
- Đồng thời, tác giả cũng giới thiệu nhiều loại sử dụng Lama, đặc biệt Big-Lama.

Tài liệu tham khảo

- [1] G. D. Bergland. “A guided tour of the fast Fourier transform”. In: *IEEE Spectrum* 6.7 (1969), pp. 41–52. DOI: 10.1109/MSPEC.1969.5213896.
- [2] Marcelo Bertalmío et al. “Image inpainting”. In: Jan. 2000, pp. 417–424.
- [3] Lu Chi, Borui Jiang, and Yadong Mu. “Fast Fourier Convolution”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 4479–4488. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/2fd5d41ec6cfab47e32164d5624269b1-Paper.pdf.
- [4] Alexei A. Efros and William T. Freeman. “Image Quilting for Texture Synthesis and Transfer”. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 341–346. ISBN: 158113374X. DOI: 10.1145/383259.383296. URL: <https://doi.org/10.1145/383259.383296>.
- [5] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].
- [6] Christine Guillemot and Olivier Le Meur. “Image Inpainting : Overview and Recent Advances”. In: *Signal Processing Magazine, IEEE* 31 (Jan. 2014), pp. 127–144. DOI: 10.1109/MSP.2013.2273004.
- [7] L Haritha and C A Prajith. “Image Inpainting Using Deep Learning Techniques: A Review”. In: *2023 International Conference on Control, Communication and Computing (ICCC)*. 2023, pp. 1–6. DOI: 10.1109/ICCC57789.2023.10165271.
- [8] Kamyar Nazeri et al. *EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning*. 2019. arXiv: 1901.00212 [cs.CV].
- [9] Deepak Pathak et al. *Context Encoders: Feature Learning by Inpainting*. 2016. arXiv: 1604.07379 [cs.CV].
- [10] Riya Shah, Anjali Gautam, and Satish Kumar Singh. “Overview of Image Inpainting Techniques: A Survey”. In: *2022 IEEE Region 10 Symposium (TENSYMP)*. 2022, pp. 1–6. DOI: 10.1109/TENSYMP54529.2022.9864513.
- [11] Tammineni Shanmukhaprasanthi et al. “A Comprehensive Study of Image Inpainting Techniques with Algorithmic approach”. In: *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*. 2023, pp. 1–5. DOI: 10.1109/ISCON57294.2023.10112205.
- [12] Roman Suvorov et al. *Resolution-robust Large Mask Inpainting with Fourier Convolutions*. 2021. arXiv: 2109.07161 [cs.CV].
- [13] Hanyu Xiang et al. “Deep learning for image inpainting: A survey”. In: *Pattern Recognition* 134 (2023), p. 109046. ISSN: 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2022.109046>. URL: <https://www.sciencedirect.com/science/article/pii/S003132032200526X>.
- [14] Jiahui Yu et al. *Generative Image Inpainting with Contextual Attention*. 2018. arXiv: 1801.07892 [cs.CV].
- [15] Haitian Zheng et al. *CM-GAN: Image Inpainting with Cascaded Modulation GAN and Object-Aware Training*. 2022. arXiv: 2203.11947 [cs.CV].