

TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐẠI HỌC QUỐC GIA TP.HCM
BỘ MÔN VIỄN THÔNG



ĐỒ ÁN HỌC KÌ 202

Tạo màu cho ảnh xám sử dụng mạng đối nghịch tạo sinh có điều kiện

GVHD: PGS.TS. Hà Hoàng Kha (hhkha@hcmut.edu.vn)

Sinh viên: Nguyễn Thành Trung - 1814515
(trung.nguyendx@hcmut.edu.vn)

Lời cảm ơn

Trước tiên, em muốn bày tỏ lòng biết ơn chân thành tới giáo viên hướng dẫn của em, thầy **PGS.TS Hà Hoàng Kha** - “giảng viên bộ môn Viễn Thông khoa Điện–Điện Tử” đã trực tiếp giúp đỡ, hướng dẫn và cho lời khuyên trong việc nghiên cứu, thực hiện, chỉnh sửa và hoàn thiện đồ án.

Em cũng muốn gửi lời cảm ơn chân thành đến trường **Đại học Bách Khoa TP.HCM** - **Đại học Quốc gia TP.HCM**, khoa **Điện-Điện Tử**, bộ môn **Viễn Thông** đã cung cấp những kiến thức nền tảng, cách thức học tập và nghiên cứu. Thật khó để tưởng tượng rằng em sẽ tiếp cận và xử lý đề tài đồ án như thế nào nếu không có những vốn liếng quý báu đó.

Cũng không thể quên cảm ơn một số anh khoá trên cùng các bạn cùng khoá và một vài cá nhân đã giúp đỡ em rất nhiều trong quá trình tìm hiểu đề tài cũng như thực hiện khảo sát lấy số liệu cho đánh giá kết quả của đồ án.

Em xin chân thành cảm ơn tất cả.

Tp. Hồ Chí Minh, ngày 04 tháng 05 năm 2021.

Sinh Viên

Tóm tắt nội dung đồ án

Tạo màu cho ảnh xám là một chủ đề nóng hỏi và thú vị trong ngành thị giác máy tính trong khoảng hai thập kỉ vừa qua. Rất nhiều những nghiên cứu đã được thực hiện và cho rất nhiều phương pháp với những kết quả ấn tượng. Tuy việc tạo màu cho mỗi trường hợp là một bài toán có không ít lời giải vì rất nhiều màu có thể có cùng mức xám. Dẫn đến việc, với một tấm ảnh xám đầu vào, ta có thể có nhiều phương án màu lựa chọn mà vẫn có được tính chân thực. Nhưng điều đó không đồng nghĩa với việc đây là một bài toán đơn giản mà trái lại là một vấn đề nan giải.

Trong đồ án này, hai mô hình mạng đối nghịch tạo sinh có điều kiện - cGAN có bộ sinh kiến trúc Unet xây dựng bằng xương sống chuyển giao từ mạng phân loại ResNet18 và bộ phân biệt kiến trúc PatchGAN 70×70 , được triển khai và huấn luyện với tập dữ liệu COCO để xử lý vấn đề này. Chất lượng mô hình đầu tiên thông qua một cuộc khảo sát thực tế 100 người với thang điểm từ 1 (rất giả) đến 5 (rất thật). Mô hình thứ hai là mô hình được cải thiện bằng cách đưa ra thay đổi nhỏ trong hàm mất và tăng số lượng dữ liệu huấn luyện. Chất lượng của hai mô hình được so sánh định tính qua những kết quả thử nghiệm, cũng như định lượng qua hai giá trị MSE và RMSE của mỗi mô hình.

Bên cạnh đó, xây dựng một API tạo màu bằng khung phần mềm Django của Python, từ đó triển khai một ứng dụng web đơn giản để tạo màu cho ảnh bằng khung phần mềm VueJS của Javascript.



Mục lục

1 Giới thiệu	1
1.1 Tổng quan	1
1.2 Mục tiêu	2
2 Cở sở lý thuyết về ảnh số và các không gian màu	3
2.1 Giới thiệu cơ bản về ảnh số	3
2.2 Ảnh nhị phân và ảnh xám	3
2.3 Ảnh màu và một số không gian màu cho ảnh màu	4
2.3.1 Không gian màu RGB	4
2.3.2 Không gian màu L*a*b*	5
2.3.3 Lợi thế trong bài toán tạo màu của không gian màu L*a*b* so với không gian màu RGB	6
3 Những phương pháp đã được đề xuất cho bài toán tạo màu	7
3.1 Phương pháp đánh dấu màu một vài điểm ảnh rồi dùng thuật toán loang	7
3.2 Phương pháp dựa vào mẫu	8
3.3 Phương pháp học sâu	8
4 Khung mô hình mạng học sâu pix2pix	9
4.1 Mạng đối nghịch tạo sinh (GAN - Generative Adversarial Network)	9
4.1.1 Khái quát về mạng GAN	9
4.1.2 Cở sở toán học và hàm mục tiêu (mất mát) của mạng GAN	11
4.1.3 Quá trình huấn luyện mạng GAN	13
4.1.4 Sự thông minh trong ý tưởng của mạng GAN	14
4.1.5 Mạng GAN không hoàn hảo	14
4.2 Mạng tích chập Unet	15
4.2.1 Khái quát về tích chập và mạng tích chập	15
4.2.2 Kiến trúc Unet	18
4.3 Kiến trúc pix2pix	19
4.3.1 Khái quát mạng GAN trong pix2pix	20
4.3.2 Bộ phân biệt Patch trong pix2pix	20
4.3.3 Hàm mục tiêu (mất mát) pix2pix	22
5 Kiến trúc mô hình cho bài toán tạo màu dựa vào khung mô hình pix2pix	24
5.1 Phát biểu bài toán, cách áp dụng vào khung mô hình pix2pix và hàm mất mát . .	25
5.2 Kiến trúc bộ sinh pix2pix và bộ sinh học chuyển giao từ xương sống ResNet18 .	26



6 Triển khai mô hình	27
6.1 Ngôn ngữ lập trình và thư viện chính	27
6.2 Tập dữ liệu, chuẩn hoá và làm giàu dữ liệu	27
6.3 Huấn luyện mô hình	28
7 Thủ nghiệm và đánh giá mô hình	31
7.1 Mô hình GAN với bộ sinh kiến trúc pix2pix	31
7.2 Mô hình GAN với bộ sinh có xương sống ResNet18	32
7.3 So sánh bộ sinh có xương sống ResNet18 trước và sau khi huấn luyện đối nghịch	33
7.4 Khảo sát đánh giá định tính chất lượng mô hình	34
7.5 So sánh kết quả sau khi huấn luyện lại với tập dữ liệu lớn hơn	39
7.6 Đánh giá định lượng hai mô hình trước và sau khi cải thiện bằng MSE và RMSE	40
8 Ứng dụng	41
8.1 Giao diện lập trình ứng dụng (API - Application Programming Interface) bằng khung phần mềm Django của Python	41
8.2 Ứng dụng web với khung phần mềm VueJS của Javascript và API	42
9 Tổng kết	43
9.1 Kết luận	43
9.2 Hạn chế và hướng cải thiện mô hình	43
9.3 Hướng phát triển đề tài	44

Danh sách hình

1.1.1 Nải chuối có thể màu lục hoặc màu vàng.	1
1.2.1 Hình ảnh ngày lễ tạ ơn được chụp năm 1911, được tái tạo màu bằng DeOldify - một dự án học sâu [23, 49].	2
2.1.1 Không gian toạ độ của ảnh số.	3
2.2.1 Ảnh nhị phân 8×8 và biểu diễn ma trận của nó.	4
2.2.2 256 mức màu của ảnh xám.	4
2.3.1 Phối trộn màu bổ sung.	5
2.3.2 Các kênh màu đỏ, lục và lam được tách ra riêng biệt [56].	5
2.3.3 Không gian màu Lab.	5
2.3.4 Các giá trị L , a và b được tách ra riêng biệt [56].	6
3.1.1 Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh [42].	7
3.2.1 Mô tả ý tưởng thuật toán ảnh có bố cục tương tự [42].	8
4.1.1 Ảnh mặt người sinh ra bởi StyleGAN [25].	10
4.1.2 Minh họa bộ sinh và bộ phân biệt trong mạng GAN [14].	10
4.1.3 Kiến trúc mạng đối nghịch tạo sinh.	11
4.1.4 Đồ thị hàm $\log_b x$	12
4.1.5 Mô phỏng quá trình huấn luyện mạng GAN [52, 53].	13
4.1.6 GAN không có khả năng sinh ra được phân phối (hàng dưới) gần giống với phân phối được chọn (hàng trên) [37].	15
4.2.1 Minh họa tích chập giữa hai hàm số f và g	15
4.2.2 Minh họa tính tích chập với bộ lọc kích thước 3×3	16
4.2.3 Sự khác biệt giữa hai quá trình học máy và học sâu trong việc xử lý ảnh.	17
4.2.4 (a) Ảnh có nhiễu. (b) Ảnh sau khi được giảm nhiễu bằng bộ lọc 3×3 . (b) Ảnh sau khi được giảm nhiễu bằng bộ lọc 7×7 [62].	17
4.2.5 Kiến trúc mô hình mạng Unet [54].	18
4.2.6 Sự khác biệt của bề mặt hàm mục tiêu khi không (a) và có (b) sử dụng kết nối tắt [33].	19
4.3.1 Mô hình pix2pix thêm màu sắc cho các nét vẽ để tạo ảnh màu. Một bài toán tương tự với bài toán tạo màu cho ảnh [22].	19
4.3.2 Tổng quát kiến trúc mã hoá-giải mã và mạng Unet [22].	20
4.3.3 Hình ảnh giả sử được đầu vào chia thành $k \times k$ Patch [30].	21
4.3.4 Mô tả vùng nhận thức qua các tầng.	21
4.3.5 Sơ đồ tổng quát kiến trúc các lớp mạng tích chập áp dụng trên một Patch 70×70	22
5.0.1 Các ứng dụng của khung mô hình pix2pix [22].	24
5.2.1 Một kiểu mạng Unet được khởi tạo theo phong cách ResNet [57].	26



6.2.1 Hình ảnh trích từ tập dữ liệu COCO.	28
6.2.2 Minh họa việc lật đổi xứng theo trực tung.	29
6.3.1 Giá trị các thành phần mất mát của bộ phân biệt và bộ sinh qua từng epoch.	29
7.1.1 Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản sau epoch thứ 34 trên tập dữ liệu COCO.	31
7.2.1 Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản có xương sống ResNet18 sau epoch thứ 15 trên tập dữ liệu COCO (không nằm trong tập huấn luyện).	32
7.2.2 Mô hình hoàn toàn bế tắc trước logo kỷ niệm 60 năm của trường ĐH Bách Khoa TP.HCM.	32
7.2.3 Trường hợp đặc biệt khi mô hình làm việc tệ với ảnh có đặc trưng đơn giản.	33
7.3.1 Những tấm ảnh đen trắng được lựa chọn để so sánh sự khác biệt.	33
7.3.2 Phía trên và dưới lần lượt là kết quả của bộ sinh trước và sau khi huấn luyện đối nghịch.	34
7.4.1 Một số kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc.	35
7.4.2 Thêm Một vài kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc.	36
7.4.3 Phần mô tả của bài đánh giá.	36
7.4.4 Phần câu hỏi và số lượng người tham gia cuộc khảo sát thực hiện thông qua Google Forms.	37
7.4.5 Những ảnh có điểm trung bình cao nhất.	37
7.4.6 Những ảnh có điểm trung bình thấp nhất.	37
7.5.1 So sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại.	39
7.5.2 Thêm một vài so sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại.	39
8.1.1 Bảo mật CORS cản trở khi thử nghiệm API	41
8.1.2 Phản hồi trả về của endpoint	41
8.1.3 Thử nghiệm API trên phần mềm Postman	42
8.2.1 Ứng dụng web tạo màu đơn giản bằng VueJS kết hợp API	42
9.3.1 Ảnh Trường ĐH Bách Khoa TP.HCM ngày xưa khi chưa có màu (trên) và sau khi tô màu: mô hình ban đầu (trái), mô hình sau khi cải thiện (phải)	44



Danh sách bảng

7.4.1 Kết quả bình chọn cho các ảnh.	38
7.4.2 Điểm trung bình của các ảnh.	38
7.6.1 Kết quả MSE và RMSE của hai mô hình trên 4 ảnh được chọn.	40

Danh sách quy ước ký hiệu và tên gọi được sử dụng trong đồ án

x, y, M, N	các số vô hướng, hoặc là một biến tổng quát tuỳ từng trường hợp
\mathbf{x}, \mathbf{y}	các vector cột
\mathbf{X}, \mathbf{Y}	các ma trận, hoặc tensor tuỳ từng trường hợp
\mathbf{X}, \mathbf{Y}	các tập hợp
$\mathbf{X}^{M \times N}$	tập hợp các ma trận kích thước M hàng, N cột với các phần tử thuộc tập \mathbf{X}
\mathbb{N}^*	tập hợp các số tự nhiên khác 0
\mathbb{R}	tập hợp các số thực
\mathbb{R}^n	tập hợp các vector n chiều với các phần tử thuộc tập \mathbb{R}
$\{x, y\}$	tập hợp chứa các phần tử x, y
$[x, y]$	tập hợp $\{x, x+1, \dots, y-1, y\}$
(x, y)	tập hợp $[x, y]$ bỏ đi phần tử x và y , hoặc toạ độ vị trí tuỳ từng trường hợp
$ \mathbf{X} $	số phần tử của tập \mathbf{X}
\sim	tương đương, thuộc về
$xy, x \times y$	tích của hai vô hướng
$x/y, \frac{x}{y}$	thương của hai vô hướng, x chia y
$\lfloor x \rfloor$	làm tròn xuống của x
$\sum_{i=1}^n x_i$	tổng của $x_1 + x_2 + \dots + x_n$
$\mathbf{O}_{m \times n}$	ma trận kích thước m hàng n cột với tất cả các phần tử bằng 0
$\mathbf{J}_{m \times n}$	ma trận kích thước m hàng n cột với tất cả các phần tử bằng 1
$\mathbf{X} \times \mathbf{Y}$	tích Descartes của tập \mathbf{X} và \mathbf{Y}
$\prod_{i=1}^n \mathbf{X}_i$	tích Descartes của n tập
\mathcal{G}, f, g	ánh xạ, hàm số
\mathcal{I}	ánh số
$\mathcal{I}^{(c)}$	kênh c của ánh số
$\arg \min_x f$	giá trị của x để hàm số f đạt giá trị nhỏ nhất.
$\arg \max_x f$	giá trị của x để hàm số f đạt giá trị lớn nhất.
e	hằng số Euler - hằng số Napier
$\log(\cdot)$	hàm logarit cơ số e
$\mathbb{E}\{\cdot\}$	kỳ vọng toán
$\widehat{(\cdot)}$	giá trị dự đoán của mô hình
$(\cdot)_{ij}, (\cdot)_{i,j}$	tất cả các giá trị của ma trận hoặc tensor ở toạ độ (i, j)
	lấy trong từng không gian bỏ đi 2 chiều sâu nhất
$(\cdot)_{i:i+m, j:j+n}$	tất cả các giá trị của ma trận hoặc tensor ở các toạ độ $[i, i+m] \times [j, j+n]$
	lấy trong từng không gian bỏ đi 2 chiều sâu nhất

Chương 1

Giới thiệu

1.1 Tổng quan

Nhu cầu phục chế ảnh màu từ ảnh xám (đen trắng) không phải là hiếm. Thông thường là ảnh do tác động của thời gian làm mất hoặc sai màu. Đôi khi lại là ảnh được chụp ở thời xưa, thời đại chưa có công nghệ chụp ảnh màu. Và vì nhiều mục đích cho việc trải nghiệm chân thực, con người chúng ta muốn tái tạo lại màu cho những hình ảnh chưa có màu đó.

Hiện nay, hầu hết những người thợ chỉnh sửa ảnh đều sử dụng phương pháp thủ công để phục hồi màu. Tuy có sự giúp đỡ của các công cụ hỗ trợ như các phần mềm thao tác với ảnh trên máy tính, để có thể có một tấm ảnh đẹp cũng đòi hỏi mất nhiều thời gian [50]. Đôi khi có thể lên đến một hoặc hai tháng đối với những bức ảnh có nhiều chi tiết phức tạp.

Ở góc nhìn kỹ thuật, phục chế màu, hay tạo màu cho ảnh xám là một bài toán phức tạp có nhiều lời giải, vì việc một đối tượng trong một bức ảnh có thể có nhiều màu thích hợp. Chẳng hạn như một nải chuối có thể màu lục khi chưa chín nhưng lại cũng có thể là màu vàng khi chín, và thậm chí có thể đã chín nhưng vẫn còn màu lục (hình 1.1.1). Do đó, việc khôi phục lại đúng chính xác màu cho một bức ảnh bất kì là chuyện gần như bất khả thi và cũng không cần thiết. Nhưng cũng không vì thế mà ta có thể gán một màu bất kì cho một đối tượng nào đó, giả sử như việc màu da người là màu xanh biển hoặc mặt trời là màu xanh lá. Nhìn một tấm ảnh sau khi được khôi phục màu lại có màu không phù hợp, phản ánh không khớp với những gì chúng ta quan sát thường ngày, hiển nhiên không phải là mục đích để chúng ta phải cực khổ đi khôi phục màu.



Hình 1.1.1: Nải chuối có thể màu lục hoặc màu vàng.

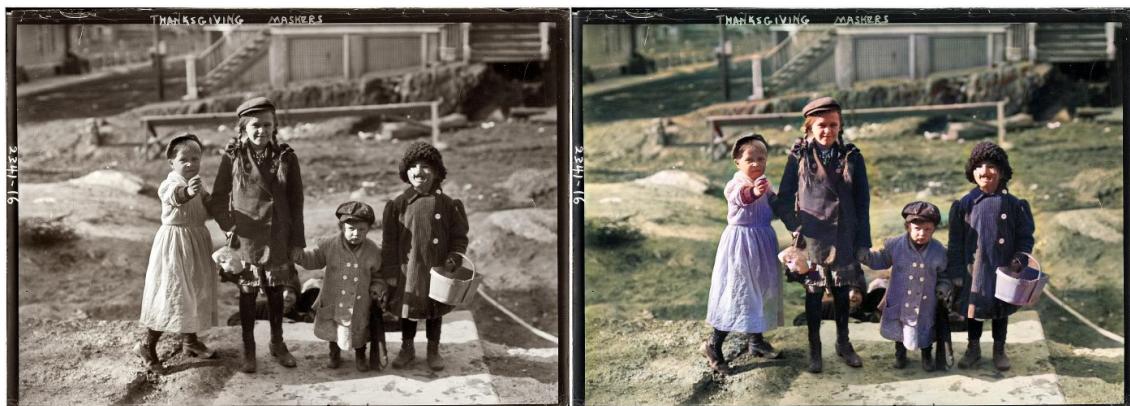
Trước khi có sự xuất hiện của ngành học máy/học sâu [78, 73] trong việc xử lý ảnh, đã có rất nhiều kỹ thuật được đưa ra cho quá trình tự động phục hồi màu cho ảnh xám [3.1, 3.2]. Nhưng để có được một kết quả ưng ý, đều hoặc nhiều hoặc ít dựa vào sự can thiệp của bàn tay con người. Sau khi học sâu bùng nổ trong ngành thị giác máy tính, rất nhiều mô hình học sâu ứng dụng mang tính chấp [4.2.1] đã có thể học và tạo ra màu cho những ảnh xám [3.3, 9.2] (hình 1.2.1).

1.2 Mục tiêu

Theo những gì đã nghiên cứu về đề tài [1.1], đồ án này được đưa ra với mục đích xây dựng và huấn luyện một mô hình mạng học sâu để tạo màu cho ảnh xám. Yêu cầu của màu tạo ra không nhất thiết phải giống hoàn toàn so với màu thực (màu của ảnh/vật gốc nếu có). Thay vào đó là độ chân thực, hợp lý của ảnh sau khi ảnh được kết hợp với màu được tạo sẽ là thứ được hướng đến. Mức độ chân thực, hợp lý sẽ được đánh giá định tính bằng một cuộc khảo sát trực tuyến gồm 100 người tham gia [7.4], và một chút định lượng thông qua hai giá trị MSE và RMSE [7.6]

Các nội dung trong đồ án sẽ được trình bày từ những cơ sở lý thuyết để giải quyết bài toán tạo màu. Từ những khái niệm nền tảng về ảnh số, cho tới kiến trúc mô hình được sử dụng. Tiếp theo là phần triển khai và huấn luyện mô hình. Sau đó là đánh giá, cải thiện mô hình và so sánh, rồi ứng dụng mô hình vừa hoàn thành. Cuối cùng là tổng kết và nêu ra hướng phát triển cho đề tài đồ án. Trong các nội dung được trình bày, những phần chính yếu, cần nhiều sự quan tâm hơn cả sẽ là:

- Lựa chọn không gian màu thích hợp để sử dụng cho bài toán tạo màu [2.3.3].
- Mạng đối nghịch tạo sinh [4.1].
- Mạnh tích chập Unet [4.2].
- Kiến trúc khung mô hình mạng học sâu pix2pix [4.3].
- Kiến Mô hình mạng học sâu xử lý bài toán tạo màu dựa vào kiến trúc pix2pix [5].
- Triển khai mô hình sử dụng ngôn ngữ lập trình Python và thư viện Pytorch [6].
- Đánh giá, cải thiện mô hình và so sánh [7].



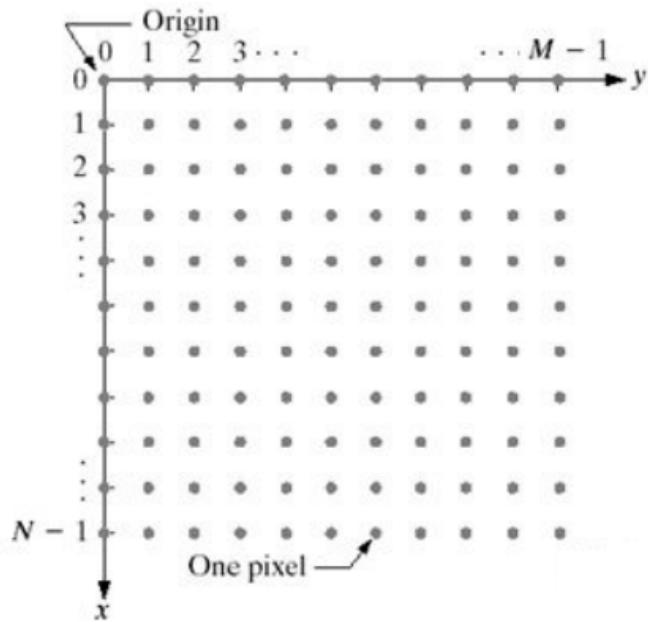
Hình 1.2.1: Hình ảnh ngày lễ tạ ơn được chụp năm 1911, được tái tạo màu bằng DeOldify - một dự án học sâu [23, 49].

Chương 2

Cơ sở lý thuyết về ảnh số và các không gian màu

2.1 Giới thiệu cơ bản về ảnh số

Máy tính là một công cụ rất mạnh mẽ để xử lý những vấn đề của con người. Nhưng chúng bị hạn chế khi chỉ xử lý dữ liệu dưới dạng các con số. Hình ảnh cũng không phải ngoại lệ. Ảnh số - ảnh mà máy tính hiểu, thực tế là tập hợp hữu hạn các điểm ảnh dưới dạng một ma trận 2 chiều $\mathcal{I} \in \mathbf{G}^{W \times H}$; $\mathbf{G} = \prod_{i=1}^C \mathbf{G}_i$. Trong đó, $W \in \mathbb{N}^*$ và $H \in \mathbb{N}^*$ lần lượt là chiều rộng và chiều cao của ảnh được tính bằng đơn vị điểm ảnh [43]. Mỗi điểm ảnh $p_{xy} \in \mathbf{G}$ là một phần tử của ảnh tại tọa độ nguyên (x, y) ; $0 \leq x \leq H - 1, 0 \leq y \leq W - 1$ (hình 2.1.1, $H = N, W = M$), có giá trị trong một tập hữu hạn ($|\mathbf{G}| \neq \infty$), biểu thị cho mức xám hoặc màu nhất định của nó. Mức xám hoặc màu nhất định được biểu diễn như thế nào tùy thuộc vào loại ảnh số: **ảnh nhị phân**, **ảnh xám (đen trắng)** [2.2] hay **ảnh màu** [2.3].



Hình 2.1.1: Không gian tọa độ của ảnh số.

2.2 Ảnh nhị phân và ảnh xám

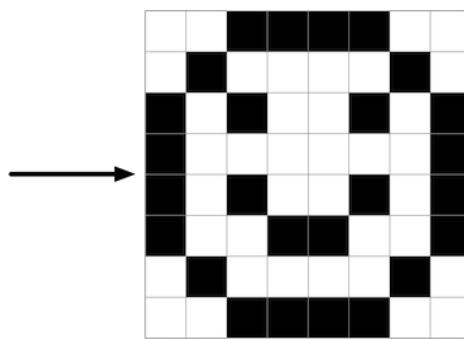
Ảnh nhị phân [67] là loại ảnh sơ khai và đơn giản nhất. Mỗi điểm ảnh $p_{xy} \in \mathbf{G}_{\text{nhi phan}} = \{0, 1\}$ được biểu diễn bằng 1 bit. Với $p_{xy} = 0$ thì điểm ảnh nhận màu đen, và ngược lại thì sẽ nhận màu trắng (hình 2.2.1).

Khác một chút so với ảnh nhị phân, ảnh xám [75] có tập giá trị đa dạng hơn là $\mathbf{G}_{\text{xam}} = [0, 255]$, được biểu diễn bằng 1 byte (8 bit). Cũng như ảnh nhị phân, giá trị điểm ảnh càng lớn thì màu của điểm ảnh đó càng gần màu trắng. Nhờ có nhiều mức màu hơn so với ảnh nhị phân (hình 2.2.2), ảnh xám cho ra những hình ảnh đẹp hơn.

```

1 1 0 0 0 0 1 1
1 0 1 1 1 1 0 1
0 1 0 1 1 0 1 0
0 1 1 1 1 1 1 0
0 1 0 1 1 0 1 0
0 1 1 0 0 1 1 0
1 0 1 1 1 1 0 1
1 1 0 0 0 0 1 1

```



Hình 2.2.1: Ảnh nhị phân 8×8 và biểu diễn ma trận của nó.

256 levels

Hình 2.2.2: 256 mức màu của ảnh xám.

2.3 Ảnh màu và một số khong gian màu cho ảnh màu

Khi nói về một con số là tốc độ của một vật, ta có thể tưởng tượng theo nhiều hướng khác nhau. Giả sử con số ta đang đề cập là 40. Nếu như ta chạy xe máy trong thành phố, thì tốc độ 40 ki-lô-mét/giờ là một vận tốc được cho phép. Nhưng nếu nó là 40 dặm/giờ (≈ 64.3738 ki-lô-mét/giờ), thì ta cũng có thể bị bắn tốc độ ngay cả khi là ở ngoài khu vực đông dân cư. Còn nếu là 40 mét/giây (= 144 ki-lô-mét/giờ), chắc chỉ có thể thấy vận tốc này ở trong các trường đua xe.

Rõ ràng, khi nói tới vận tốc, con số là phần quan trọng. Nhưng đồng thời, đơn vị biểu diễn là điều buộc phải có. Ảnh màu cũng vậy, sẽ là khá thiếu sót nếu ta cần xử lý sâu một tấm ảnh nhưng không biết được ảnh đang được xử lý đang thuộc khong gian màu nào. Vì không tồn tại duy nhất một khong gian màu [70] cho mọi loại ảnh. Điều này là do nhu cầu và mục đích của con người rất đa dạng, dẫn tới có nhiều loại khong gian màu đã được tạo ra để đáp ứng.

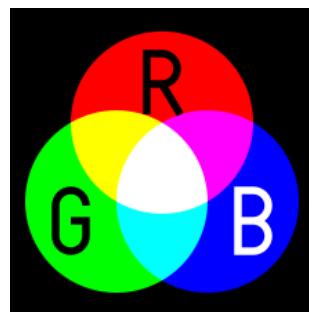
Các khong gian biểu diễn ảnh màu đều có nhiều hơn 1 kênh so với ảnh nhị phân và ảnh xám, và thường là $C = 3$ kênh giá trị để biểu diễn cho một điểm ảnh ($p_{xy} \in \mathbf{G}_1 \times \mathbf{G}_2 \times \mathbf{G}_3$). Giống với ảnh xám, mỗi kênh màu đều sử dụng 1 byte số nguyên. Trong đồ án này, để tiện lưu trữ và xử lý, giá trị trong mỗi điểm ảnh sẽ được tách riêng thành C ma trận kích thước $W \times H$, xếp chồng thành một tensor¹ có số chiều $C \times W \times H$ [4.2.1, 6].

2.3.1 Không gian màu RGB

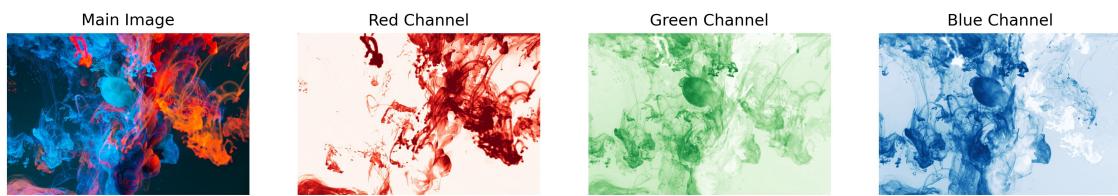
Hầu hết khi làm việc với ảnh số, ta sẽ thao tác với ảnh có không gian màu là **RGB** [81], ảnh có 3 kênh màu đỏ-lục-lam (hình 2.3.2) ($\mathbf{G}_{\text{RGB}} = \mathbf{G}_R \times \mathbf{G}_G \times \mathbf{G}_B$). Đây cũng là không gian màu phổ biến với đại đa số mọi người và được sử dụng rộng rãi hiện nay. Các kênh màu \mathbf{G}_R , \mathbf{G}_G , \mathbf{G}_B của không gian màu **RGB** có giá trị nằm trong đoạn [0, 255].

Chỉ với 3 màu đỏ, lục và lam, nhưng ta lại có thể tạo ra được khoảng hơn 16 triệu (chính xác là 256^3) màu khác nhau. Trong đó, 7 màu chủ đạo là đỏ, lục, lam, trắng, vàng, xanh ngọc, và hồng (hình 2.3.1). Thêm đỏ vào xanh lá cây tạo ra vàng. Thêm vàng vào xanh lam tạo ra

¹Wikipedia, “Tensor”, <https://en.wikipedia.org/wiki/Tensor>



Hình 2.3.1: Phối trộn màu bổ sung.

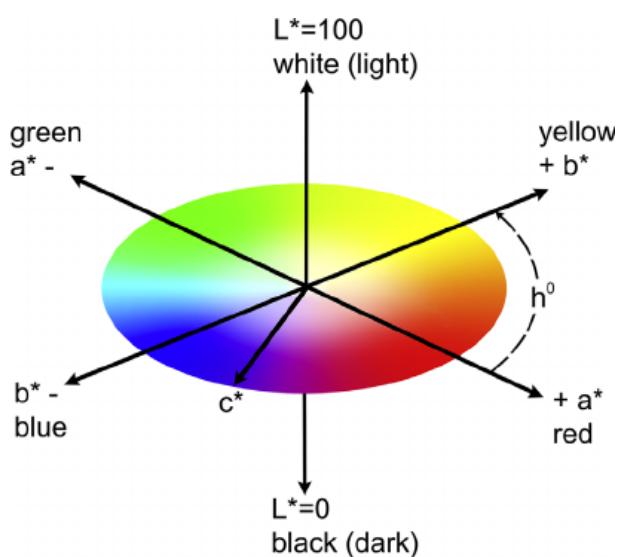


Hình 2.3.2: Các kênh màu đỏ, lục và lam được tách ra riêng biệt [56].

trắng. Sự kỳ diệu này có được dựa trên cơ sở phản ứng sinh lý học của mắt người với ánh sáng [81].

Với nhu cầu hình ảnh ghép, đã xuất hiện phương án thêm vào 1 kênh 8 bit cho độ trong suốt. Kênh trong suốt được biết đến với tên là *alpha*, vì thế, nên định dạng này có tên là **RGBA** [82]. Cũng lưu ý rằng, vì nó không thay đổi bất kỳ cái gì trong mô hình **RGB**, nên **RGBA** không phải là một mô hình màu khác biệt. Nó chỉ là định dạng bổ sung thêm thông tin về độ trong suốt cùng với thông tin màu.

2.3.2 Không gian màu L*a*b*



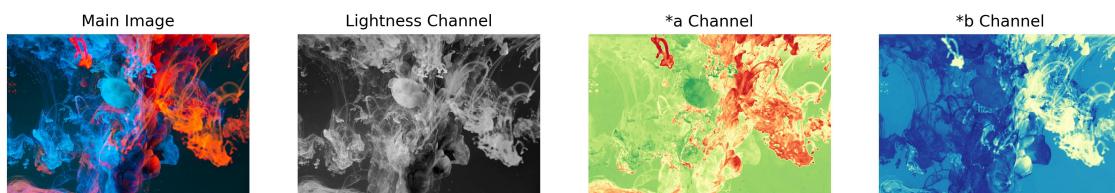
Hình 2.3.3: Không gian màu Lab.

Ngoài **RGB**, một không gian màu khác cũng được sử dụng khá nhiều là không gian màu **L*a*b*** [68] - hệ màu được xây dựng dựa trên khả năng cảm nhận của mắt người. Các giá trị của **L*a*b*** mô tả tất cả những màu mà mắt người bình thường có thể nhìn thấy được. Mô hình màu này được xem là độc lập với thiết bị và thường được sử dụng như một cơ sở tham chiếu khi chuyển đổi từ một không gian màu này sang một không gian màu khác.

Cũng như **RGB**, Không gian **L*a*b*** cũng quan tâm đến 3 thông số của mỗi điểm ảnh (hình 2.3.4) ($G_{L^*a^*b^*} = G_{L^*} \times G_{a^*} \times G_{b^*}$). Thông số đầu tiên là **L*** đại diện cho độ sáng của của mỗi điểm ảnh. Giá trị **L*** càng lớn thì điểm ảnh càng nghiêng về màu trắng, ngược lại thì sẽ là màu đen. Hai thông số còn lại là **a*** và **b*** sẽ mang thông tin lần lượt là lục-đỏ và vàng-lam. Giá trị

a^* càng thấp thì điểm ảnh nghiêng về lục nhiều, đỏ ít. Ngược lại thì lục ít, đỏ nhiều và tương tự với giá trị b^* . Theo mô hình $L^*a^*b^*$, tất cả các màu có cùng độ sáng L^* sẽ cùng nằm trên một mặt phẳng có dạng hình tròn theo 2 trục a^* và b^* (hình 2.3.3).

Giá trị của kênh L^* nằm trong tập $G_{L^*} = [0, 100]$. Về phần 2 kênh còn lại là a^* và b^* , không có cụ thể một khoảng nhất định mà tùy thuộc vào phần mềm, chương trình mà ta sử dụng. Nhưng thông thường sẽ là $G_{a^*} \times G_{b^*} = [-128, 127] \times [-128, 127]$.



Hình 2.3.4: Các giá trị L , a và b được tách ra riêng biệt [56].

2.3.3 Lợi thế trong bài toán tạo màu của không gian màu $L^*a^*b^*$ so với không gian màu RGB

Ngoài hai không gian màu **RGB** và **$L^*a^*b^*$** , còn một số không gian màu khác như **CMYK** [69], **HSV** [76],... Vậy nên, lựa chọn một không gian màu thích hợp để xử lý một bài toán cũng là một vấn đề cần đắn đo.

Giữa hai không gian màu **RGB** và **$L^*a^*b^*$** , mặc dù **RGB** có sự phổ biến nhiều hơn, nó vẫn không được lựa chọn để xử lý cho bài toán tạo màu. Sẽ có hai lợi thế khi sử dụng **$L^*a^*b^*$** thay vì **RGB**:

- i) Không gian màu **$L^*a^*b^*$** được thiết kế để đồng nhất về cảm giác về màu sắc của con người, điều này làm cho không gian màu này trở nên lý tưởng cho việc xử lý máy tính.
- ii) Tận dụng kênh mức xám đầu vào để làm kênh độ sáng L^* bằng cách điều chỉnh tỉ lệ. Khi đó mô hình chỉ phải dự đoán 2 kênh a^* và b^* thay vì phải dự đoán 3 kênh trong không gian **RGB**.

Trong hai lợi thế được nêu ra, (ii) được cho là lý do chính mà không gian màu **$L^*a^*b^*$** trở thành không gian trọng đại đa số các bài toán tạo màu cho ảnh xám. Và trong đồ án này, cũng không phải là một trường hợp ngoại lệ, không gian màu sử dụng để xử lý bài toán sẽ là không gian **$L^*a^*b^*$** .

Chương 3

Những phương pháp đã được đề xuất cho bài toán tạo màu

Trong khoảng hai thập kỷ vừa qua, rất nhiều phương pháp đã được đề xuất. Chúng có thể được chia ra làm 3 loại phương pháp chính: **đánh dấu màu một vài điểm ảnh rồi dùng thuật toán loang, dựa vào mẫu có bối cảnh tương tự** và cuối cùng là **phương pháp học sâu**. Hai loại phương pháp đầu tiên chưa phải là một phương pháp hoàn toàn tự động, mà vẫn phải cần có sự can thiệp từ con người. Phương pháp thứ 3 là sử dụng học sâu thì hoàn toàn tự động, khi mô hình học sâu có thể học được màu tương ứng của các đối tượng từ những dữ liệu huấn luyện.

3.1 Phương pháp đánh dấu màu một vài điểm ảnh rồi dùng thuật toán loang

Được đề xuất bởi Levin [32], từ một tấm ảnh xám đầu vào, ta sẽ vẽ một vài màu cơ bản từ đó làm nền tảng, định hướng cho mô hình. Ý tưởng này xuất phát từ quan sát rằng, những điểm ảnh có độ sáng gần bằng nhau (mức xám xấp xỉ) và có khoảng cách trên ảnh gần nhau sẽ có khả năng tương tự cao dẫn đến màu giống nhau.

Một vài cách cải thiện độ hiệu quả của phương pháp trên như là sử dụng thông tin các cạnh để thuật toán loang hoạt động chính xác hơn [19]. Hay theo như Luan [34], ta sẽ đánh dấu mỗi màu cho mỗi phân đoạn khác nhau (hình 3.1.1).



Hình 3.1.1: Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh [42].

Phương pháp thuộc loại này nhìn chung có kết quả màu rất tốt, nhưng lại đòi hỏi không ít sức người. Đặc biệt với những ảnh nhiều chi tiết thì, việc đánh dấu màu sẽ trở nên khó khăn và tốn nhiều thời gian.Thêm nữa, việc chọn màu để đánh dấu cũng không phải là một vấn đề dễ dàng với một người có ít kinh nghiệm về màu sắc.

3.2 Phương pháp dựa vào mẫu

Đây là một loại phương pháp sử dụng một mẫu giống hoặc tương tự để giải quyết bài toán. Welsh [65] khởi xướng phương pháp này bằng cách dựa vào màu của ảnh có bối cảnh tương tự (hình 3.2.1). Ta chọn ra một tấm ảnh mẫu có bối cảnh tương tự rồi dựa vào màu của ảnh mẫu, kết hợp chỉnh sửa để đưa ra dự đoán. Một vấn đề gặp phải khi ta sử dụng phương pháp trên đó là sự gắn kết không gian đã gần như bị bỏ qua. Để khắc phục, Ironi [21] đề xuất việc đưa màu của những phân đoạn của ảnh mẫu sang dùng để đánh dấu màu rồi dùng thuật toán loang [3.1]. Một hướng khác, Tai [60] xây dựng xác suất từng phân đoạn của cả hai bức hình, sau đó chuyển màu từ ảnh mẫu sang ảnh cần tạo màu theo những vùng có kết quả thống kê gần tương ứng.



Hình 3.2.1: Mô tả ý tưởng thuật toán ảnh có bối cảnh tương tự [42].

Dù loại phương pháp này đã giảm thiểu được phần nhiều tính thủ công, nhưng lại bị phụ thuộc lớn vào ảnh mẫu, ảnh mà phải có bối cảnh tương tự so với ảnh cần xử lý.

3.3 Phương pháp học sâu

Sau này, bằng cách tận dụng những cặp ảnh (xám, màu), rất nhiều phương pháp dựa vào học sâu đã được ra đời. Đầu tiên nhất là Cheng [10], khi triển khai một mô hình mạng nơ-ron sâu hoàn toàn tự động tạo màu, thông qua việc xây dựng và tối ưu bài toán bình phương tối thiểu, với đầu vào của mô hình là bộ những mô tả ngữ cảnh. Nổi tiếng và thành công nhất trong phương pháp loại này có thể nói đến là Zhang [87], bằng việc học phân phối màu của mỗi điểm ảnh. Mô hình của Zhang được huấn luyện với đa thức entropy chéo kết hợp việc cân bằng các lớp hiếm để tạo ra được những màu ít xuất hiện.

Riêng với những phương pháp học sâu đời đầu như Cheng có thể cải tiến dựa vào những nghiên cứu về mạng tích chập. Về sau, khi tập dữ liệu để huấn luyện trở nên lớn thì những mô hình học sâu gần như đã có thể tự động tô màu một cách cực kì chân thật, không thua gì so với kết quả được làm thủ công bởi những người thợ chỉnh sửa ảnh lâu năm. Và có thể đánh lừa mắt người xem giữa ảnh gốc và ảnh được tô màu.

Đồ án này trình bày một phương pháp học sâu, với mô hình học sâu ứng dụng khung mô hình mạng học sâu **pix2pix** được đề xuất bởi Isola [22]. Khung mô hình sử dụng mạng **GAN** [15] có điều kiện với bộ sinh có kiến trúc mạng **Unet** [54] kết hợp với bộ phân biệt Patch.

Chương 4

Khung mô hình mạng học sâu pix2pix

Với mong muốn có một cái nhìn rõ ràng về kiến trúc mô hình được áp dụng cho bài toán tạo màu của đồ án này, sẽ là không hề lãng phí nếu ta dành một chút thời gian để đi qua những thành phần chính của khung mô hình pix2pix.

Nòng cốt của khung là một mạng đối nghịch tạo sinh GAN [4.1] với một sinh theo kiến trúc mạng Unet [4.2]. Những đặc điểm khác biệt của mạng GAN trong khung mô hình so với mạng GAN thông thường cũng sẽ được trình bày [4.3.1]. Như việc tạo nhiễu, hay điều kiện đầu vào của bộ phân biệt. Ngoài ra, bộ phân biệt có kiến trúc Patch cũng sẽ được đề cập [4.3.2].

4.1 Mạng đối nghịch tạo sinh (GAN - Generative Adversarial Network)

Yann LeCun, giám đốc Facebook AI Research, từng nhận xét về ý tưởng của mạng GAN:
“The most interesting idea in the last 10 years in Machine Learning.”

Tạm dịch:

“Là ý tưởng thú ví nhất về Học Máy trong vòng 10 năm qua.”

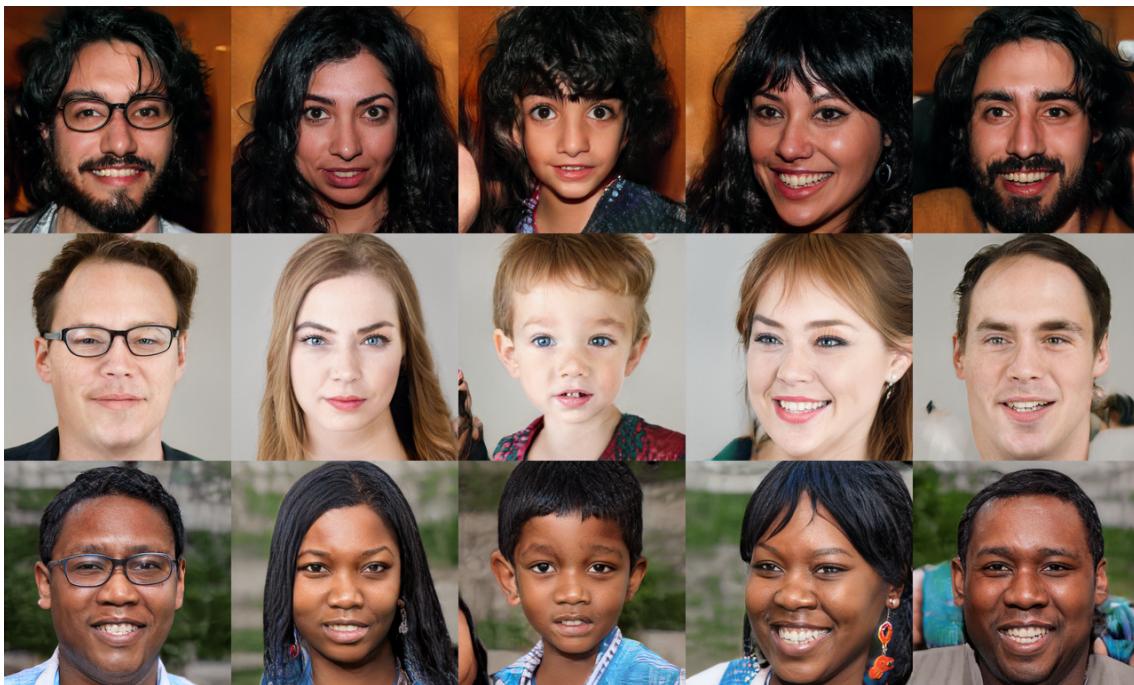
Những lời có cánh từ một trong những người có tầm ảnh hưởng trong ngành khoa học máy tính, quả là một điều tuyệt vời. Và thật vậy, GAN đã có được rất nhiều thành tựu từ khi được giới thiệu với công chúng từ năm 2014 bởi Ian J.Goodfellow và công sự của mình [15].

4.1.1 Khái quát về mạng GAN

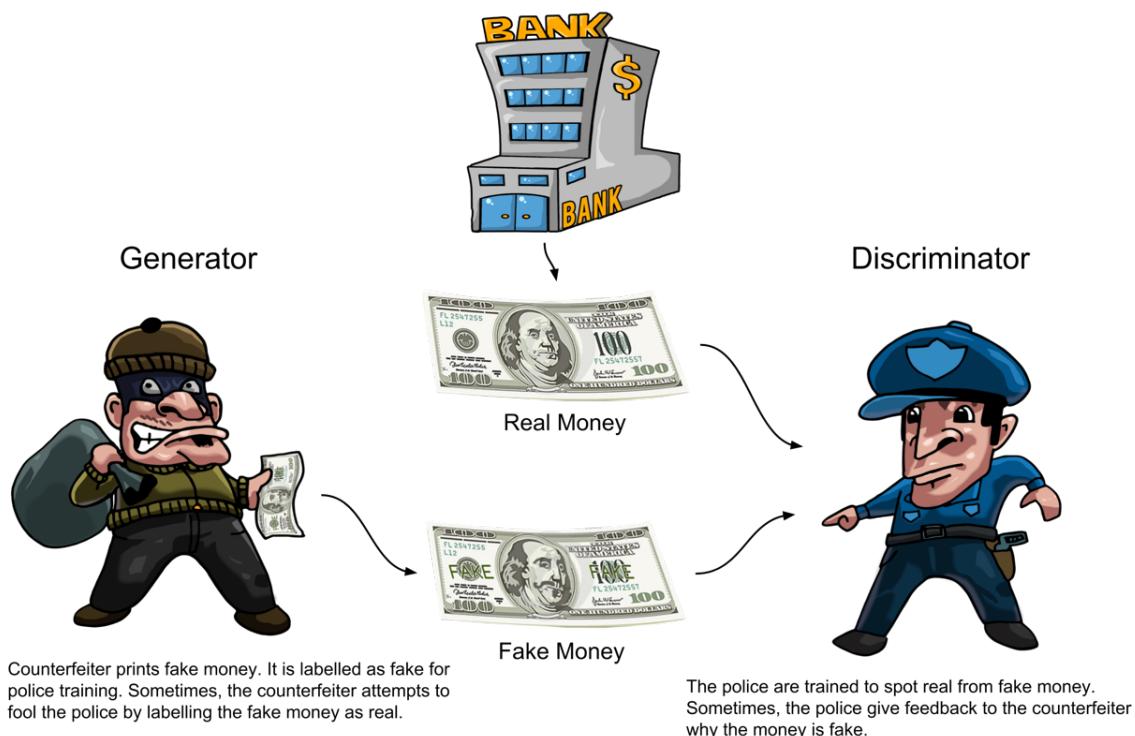
Mạng GAN thuộc nhóm mô hình sinh dữ liệu mới. Dữ liệu sinh ra nhìn như thật nhưng không phải thật. Ví dụ như ảnh mặt người (hình 4.1.1) là do GAN sinh ra, không phải mặt người thật.

G - Generative chỉ sinh, N - Network là mạng, còn A - Adversarial là đối nghịch. Lý do đối nghịch là trong mạng này được tạo nên từ sự kết hợp giữa 2 mạng là **bộ sinh** (*G - Generator*) và **bộ phân biệt** (*D - Discriminator*), luôn luôn đối nghịch nhau trong quá trình huấn luyện. Trong khi bộ sinh cố gắng sinh ra các dữ liệu giống như thật thì bộ phân biệt lại cố gắng phân biệt đâu là dữ liệu được sinh ra từ bộ sinh và đâu là dữ liệu thật.

Giả sử như bài toán đưa cho GAN là sinh ra tiền giả giống như tiền thật để có thể dùng được (hình 4.1.2), thì bộ sinh là người làm tiền giả, còn bộ phân biệt giống như cảnh sát. Người làm tiền giả sẽ cố gắng làm ra những tờ tiền giả làm sao để cảnh sát không biết đó là giả, còn cảnh sát thì cố gắng học để phân biệt được tiền nào là giả, tiền nào là thật.



Hình 4.1.1: Ảnh mặt người sinh ra bởi StyleGAN [25].



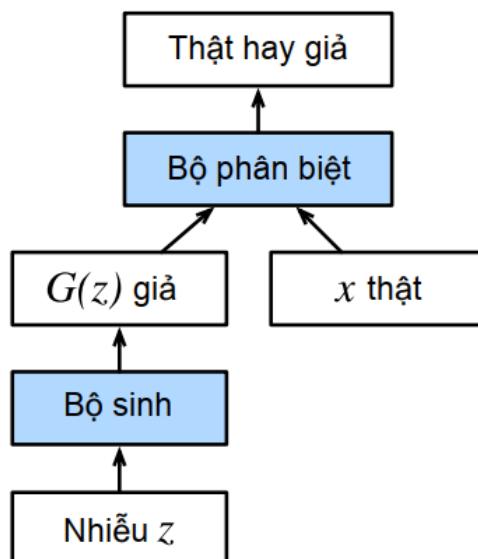
Hình 4.1.2: Minh họa bộ sinh và bộ phân biệt trong mạng GAN [14].

Mục tiêu cuối cùng của GAN là người làm tiền giả phải có khả năng làm tiền giả sao cho cảnh sát không phân biệt được đâu là thật đâu là giả (50/50) để đem tiền giả đi tiêu thụ. Trong quá trình huấn luyện mạng GAN, thì nhiệm vụ của cảnh sát là học cách phân biệt tiền giả và tiền thật, bên cạnh đó là nói cho người làm tiền giả là nên làm giả như thế nào cho giống thật hơn. Dần dần thì người làm tiền giả sẽ làm ra được tiền giống tiền thật và cảnh sát cũng trở nên thành thạo trong việc phân biệt tiền thật hay giả.

Ý tưởng của GAN bắt nguồn từ [Non-Zero-Sum Games](#)¹, là một trò chơi đối kháng giữa 2 người. Nếu một trong hai người thắng, thì người còn lại sẽ thua. Ở mỗi lượt, thì cả 2 đều muốn tối đa hóa cơ hội thắng của mình và tối thiểu hóa cơ hội thắng của đối phương. Trong lý thuyết trò chơi thì mô hình sẽ hội tụ khi cả bộ sinh và bộ phân biệt đạt tới trạng thái cân bằng Nash², tức là các bước tiếp theo của bất cứ ai trong hai người đều không làm thay đổi cơ hội thắng của ai cả.

4.1.2 Cở sở toán học và hàm mục tiêu (mất mát) của mạng GAN

Dưới góc độ toán học, một bộ sinh sẽ sinh ra dữ liệu tốt (dữ liệu mà chúng ta mong muốn) nếu ta không thể chỉ ra đâu là dữ liệu giả và đâu là dữ liệu thật. Trong thống kê, điều này được gọi là bài kiểm tra từ hai tập mẫu - một bài kiểm tra để trả lời câu hỏi liệu tập dữ liệu $\mathbf{X} = \{x_1, x_2, \dots, x_n\}$ và $\mathbf{X}' = \{x'_1, x'_2, \dots, x'_n\}$ có được rút ra từ cùng một phân phối hay không. Sự khác biệt chính giữa hầu hết những bài nghiên cứu thống kê và GAN, là GAN sử dụng ý tưởng này theo kiểu có tính xây dựng. Nói cách khác, thay vì chỉ huấn luyện mô hình để nói “này, hai tập dữ liệu đó có vẻ như không đến từ cùng một phân phối”, thì GAN sử dụng phương pháp **kiểm tra trên hai tập mẫu**³ để cung cấp tín hiệu cho việc huấn luyện cho bộ sinh. Điều này cho phép ta cải thiện bộ sinh để có thể sinh dữ liệu tối khi ra được thứ gì đó giống như dữ liệu thực. Ở mức tối thiểu nhất, nó cần phải lừa được bộ phân biệt, kể cả bộ phân biệt của ta là một mạng nơ-ron sâu tân tiến nhất.



Hình 4.1.3: Kiến trúc mạng đối nghịch tạo sinh.

Bộ phân biệt $D(\cdot)$ là một mạng học sâu phân loại nhị phân nhằm phân biệt đầu vào $\mathbf{x} \in \mathbb{R}^n$ là thật (từ dữ liệu thật) hoặc giả (từ bộ sinh) (hình 4.1.3), được học theo kiểu giám sát. Đầu ra của bộ phân biệt là một số vô hướng $o \in \mathbb{R}$ dự đoán cho đầu vào \mathbf{x} , và sẽ được đưa qua hàm sigmoid $S(x) = 1/(1 + e^{-x})$, $\mathbf{R} = (0, 1)$ để nhận được xác suất dự đoán với giá trị càng gần 1 thì bộ phân biệt càng có xu hướng quyết định đó là dữ liệu thật. Giả sử mỗi cặp dữ liệu huấn luyện thứ i có dạng $(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{0, 1\}$. Bộ phân biệt cần được tối ưu sẽ có dạng entropy

¹Stanford, ‘Non-Zero-Sum Games’, <https://stanford.io/3nCiLKq>

²Jørgen Veisdal, ‘The Nash equilibrium, explained’, <https://bit.ly/3ea57Lr>

³Wikipedia, ‘Two-sample hypothesis testing’, <https://bit.ly/3vmFwVD>

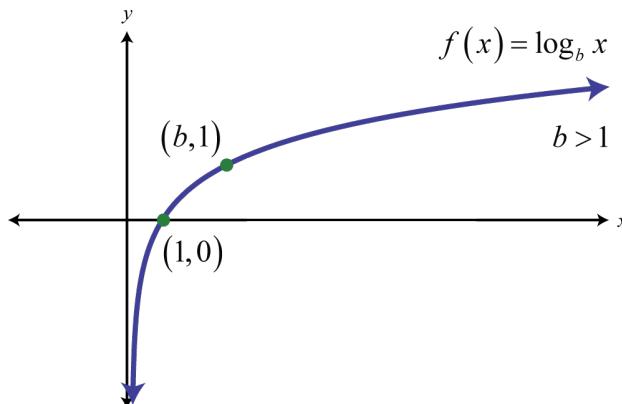
chéo [72], nghĩa là:

$$D^* = \arg \min_D \left\{ -\frac{1}{N} \sum_{i=1}^N [y_i \log D(\mathbf{x}_i) + (1 - y_i) \log (1 - D(\mathbf{x}_i))] \right\} \quad (4.1)$$

Còn đối với bộ sinh $G(\cdot)$ - cũng là một mạng học sâu, sẽ được học theo kiểu không giám sát. Trước tiên nó cần được cho vài tham số ngẫu nhiên được xem là nhiễu $\mathbf{z} \in \mathbb{R}^d$ từ một nguồn⁴, ví dụ phân phối chuẩn $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 1)$. Ta thường gọi \mathbf{z} như là một biến tiềm ẩn. Mục tiêu của mạng sinh là đánh giá bộ phân biệt để phân loại $\mathbf{x}' = G(\mathbf{z})$ là dữ liệu thật, nghĩa là, ta muốn $D(G(\mathbf{z})) \approx 1$. Một cách diễn đạt khác, cho trước một bộ phân biệt D , ta sẽ cập nhật tham số của bộ sinh G nhằm cực đại hóa mất mát entropy chéo như trong (4.1) khi $y = 0$, tức là:

$$\begin{aligned} G^* &= \arg \max_G \left\{ -\frac{1}{N_{\text{nhiều}}} \sum_{i=1}^{N_{\text{nhiều}}} (1 - y_i) \log (1 - D(G(\mathbf{z}_i))) \right\} \\ &= \arg \max_G \left\{ -\frac{1}{N_{\text{nhiều}}} \sum_{i=1}^{N_{\text{nhiều}}} \log (1 - D(G(\mathbf{z}_i))) \right\} \end{aligned} \quad (4.2)$$

Trong thực tế, công thức (4.2) không phải là một công thức tốt để tối ưu G [61]. Là bởi vì, khi mới bắt đầu huấn luyện, bộ sinh G gần như không có khả năng sinh được dữ liệu có phân phối gần giống với phân phối ta cần. Khả năng cao là ta sẽ có $D(G(\mathbf{z})) \approx 0 \Leftrightarrow (1 - D(G(\mathbf{z}))) \approx 1$. Mà ta dễ dàng thấy, độ dốc hay chính là đạo hàm của hàm $\log(x)$ giảm khi $x : 0 \rightarrow 1$ (hình 4.1.4, xét cơ số $b = e$).



Hình 4.1.4: Đồ thị hàm $\log_b x$.

Nếu như x gần giá trị 1, đạo hàm của $\log(x)$ sẽ bé. Để gây hiện tượng biến mất gradient [84], không cập nhật tham số được cho G . Thay vào đó, khi tối ưu G , ta sẽ sử dụng chiến lược cực tiểu hóa mất mát entropy chéo như trong (4.1) với $y = 1$:

$$\begin{aligned} G^* &= \arg \min_G \left\{ -\frac{1}{N_{\text{nhiều}}} \sum_{i=1}^{N_{\text{nhiều}}} y_i \log (D(G(\mathbf{z}_i))) \right\} \\ &= \arg \min_G \left\{ -\frac{1}{N_{\text{nhiều}}} \sum_{i=1}^{N_{\text{nhiều}}} \log (D(G(\mathbf{z}_i))) \right\} \end{aligned} \quad (4.3)$$

⁴Số chiều d của nhiễu đầu vào cũng có ảnh hưởng tới kết quả của mô hình GAN. Và không có một con số nào là tuyệt đối tốt nhất cho mọi kiến trúc [45].

Khi làm như vậy, ta sẽ có đạo hàm của $\log(D(G(\mathbf{z})))$ tại giá trị $D(G(\mathbf{z})) \approx 0$ là lớn hơn so với (4.2), tránh va phải việc gradient biến mất. Vì đạo hàm của $\log(x)$ khi x ở gần 0 lớn hơn khi x ở gần 1.

Tóm lại, D và G đang chơi trò “cực tiểu hoá cực đại” với một hàm mục tiêu toàn diện như sau:

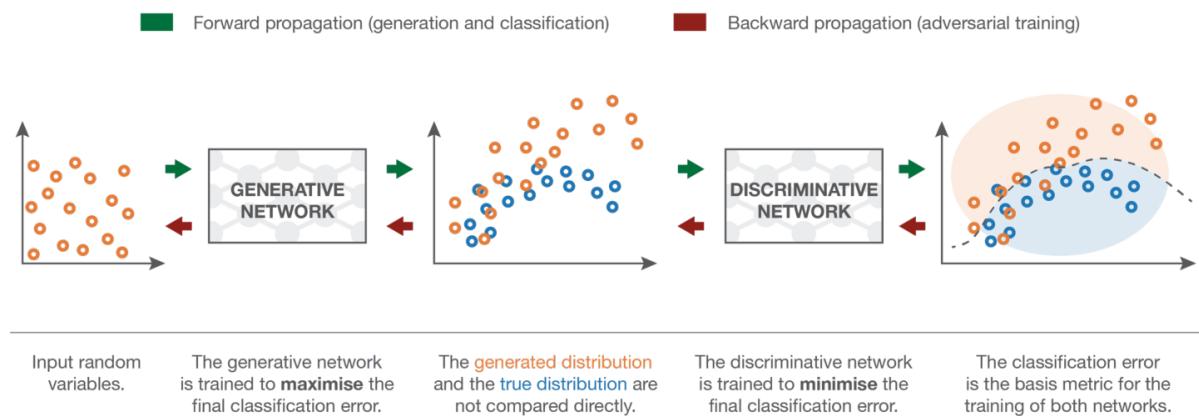
$$(D, G)^* = \arg \min_D \arg \max_G \left\{ -\mathbb{E}_{\mathbf{x} \sim \text{dữ liệu thật}} \log D(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim \text{nhiều}} \log(1 - D(G(\mathbf{z}))) \right\} \quad (4.4)$$

$$\Leftrightarrow (D, G)^* = \arg \min_G \arg \max_D \left\{ \mathbb{E}_{\mathbf{x} \sim \text{dữ liệu thật}} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim \text{nhiều}} \log(1 - D(G(\mathbf{z}))) \right\} \quad (4.5)$$

Phép toán $\mathbb{E}\{\cdot\}$ trong [(4.4), (4.5)] chính là kì vọng toán, tương đương việc lấy trung bình của tất cả dữ liệu.

Từ hàm mất mát (4.5), ta nhận thấy rằng việc huấn luyện bộ phân biệt và bộ sinh là đối nghịch nhau. Trong khi D cố gắng cực đại mất mát thì G lại đi theo hướng cực tiểu mất mát. Về mặt lý thuyết, quá trình dạy học cho GAN kết thúc khi mô hình GAN đạt đến trạng thái cân bằng của hai bộ trong mạng, tức cân bằng Nash.

4.1.3 Quá trình huấn luyện mạng GAN



Hình 4.1.5: Mô phỏng quá trình huấn luyện mạng GAN [52, 53].

Huấn luyện mạng GAN cũng không quá phức tạp nhưng lại tinh vi (hình 4.1.5) nên cần được giới thiệu. Sẽ có hai phiên: lan truyền thuận và lan truyền ngược. Ở phiên lan truyền thuận, ta chọn ngẫu nhiên b vector nhiễu từ một phân phối nhiễu xác định trước $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_b] \in \mathbb{R}^{d \times b}$ rồi tạo ảnh:

$$\begin{aligned} \mathbf{X}' &= [\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_b] \\ &= G(\mathbf{Z}) \\ &= [G(\mathbf{z}_1), G(\mathbf{z}_2), \dots, G(\mathbf{z}_b)] \in \mathbb{R}^{n \times b} \end{aligned} \quad (4.6)$$

Những ảnh được tạo ra sẽ được gán nhãn $y = 0$, kết hợp với b vector⁵ chọn từ tập dữ liệu thật $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_b] \in \mathbb{R}^{n \times b}$ được gán nhãn $y = 1$. Đầu vào của bộ phân biệt sẽ là

⁵Việc chọn số lượng mẫu từ tập dữ liệu thật nên là bằng số lượng mẫu từ dữ liệu giả, sẽ tránh được hiện tượng không cân bằng khi huấn luyện một bộ phân loại [6].

$\bar{\mathbf{X}} = [\mathbf{X}', \mathbf{X}] \in \mathbb{R}^{2b \times n}$ để phân loại. Hoành thành phiên lan truyền thuận. Kết quả phân loại là các xác suất xem mỗi điểm dữ liệu là thuộc dữ liệu thật hay là giả:

$$\begin{aligned}\hat{\mathbf{Y}}_D &= D(\bar{\mathbf{X}}) = D([\mathbf{X}', \mathbf{X}]) \\ &= [D(\mathbf{x}'_1), D(\mathbf{x}'_2), \dots, D(\mathbf{x}'_b), D(\mathbf{x}_1), D(\mathbf{x}_2), \dots, D(\mathbf{x}_b)] \\ &= [\hat{y}_{D;1}, \hat{y}_{D;2}, \dots, \hat{y}_{D;2b}] \in \mathbb{R}^{1 \times 2b}\end{aligned}\quad (4.7)$$

Sau khi có được $\hat{\mathbf{Y}}_D$, ta bắt đầu phiên lan truyền ngược bằng việc lấy nhãn $\mathbf{Y}_D = [\underbrace{0, 0, \dots, 0}_{b \text{ mẫu}}, \underbrace{1, 1, \dots, 1}_{b \text{ mẫu}}] \in \mathbb{R}^{1 \times 2b}$ để đưa vào hàm mất mát (4.1) và tối ưu bằng gradient thông qua một thuật toán tối ưu [55]. Sau khi tối ưu, bộ phân biệt sẽ có trở nên tốt hơn trong việc phân biệt hai phân phối. Một nửa của phiên lan truyền đã được hoàn thành.

Một nửa còn lại của phiên lan truyền ngược tiếp tục. Ta cần cố định các gradient của D , rồi lấy kết quả tính được trong phiên trước $\hat{\mathbf{Y}}_D = D(\mathbf{X}') = [\hat{y}_{D;1}, \hat{y}_{D;2}, \dots, \hat{y}_{D;b}] \in \mathbb{R}^{1 \times b}$, rồi cùng với nhãn $\mathbf{Y}_D = [1, 1, \dots, 1] \in \mathbb{R}^{1 \times b}$ được truyền vào hàm mất mát (4.3), và ta cũng sẽ tối ưu bằng gradient thông qua một thuật toán tối ưu. Vì trong (4.3) cũng có cùng D , nên như đã đề cập ban đầu, các gradient của D cần được cố định lại trước khi huấn luyện G . Qua mỗi lượt tối ưu, phân phối của tập ảnh sẽ được điều chỉnh cho giống với phân phối tập dữ liệu thật.

Trong suốt quá trình tối ưu, phân phối nhiều được giữ nguyên. Việc thay đổi phân phối này làm phức tạp thêm cho công cuộc huấn luyện mà cũng chưa chắc đem lại kết quả khả quan hơn.

4.1.4 Sự thông minh trong ý tưởng của mạng GAN

Thực sự mà nói, nếu ta có thể biết được cách biểu diễn cụ thể cho phân phối dữ liệu thật, thì việc sinh đã trở nên đơn giản. Tuy nhiên, vì việc đó là bất khả thi, nên những thuật toán mới được sinh ra. Nhằm để tìm ra một ảnh xạ, sao cho từ một tập đích là một phân phối đơn giản, ta có được kết quả ảnh xạ nằm trên một phân phối tương tự với phân phối dữ liệu thật - thứ mà ta không biết cụ thể.

Đã từng có một cách tiếp cận GMN (Generative Matching Networks)⁶. Phương pháp này sẽ so sánh phân phối của ảnh ánh xạ với phân phối dữ liệu thật dự vào các mẫu từ hai tập. Dẫu vậy, việc cài đặt và huấn luyện cần một phép đo hợp lý để so sánh hai phân phối là một điều không hề dễ dàng [51]. Và hay thay, GAN đưa ra một giải pháp thông minh, là mượn một mạng học sâu khác để giải quyết. Nhờ vào việc để bộ phân biệt - một mạng học sâu không quá phức tạp để tối ưu dẫn đường, công việc của ta bây giờ sẽ chỉ là tập trung nhiều cho tối ưu ảnh xạ bộ sinh.

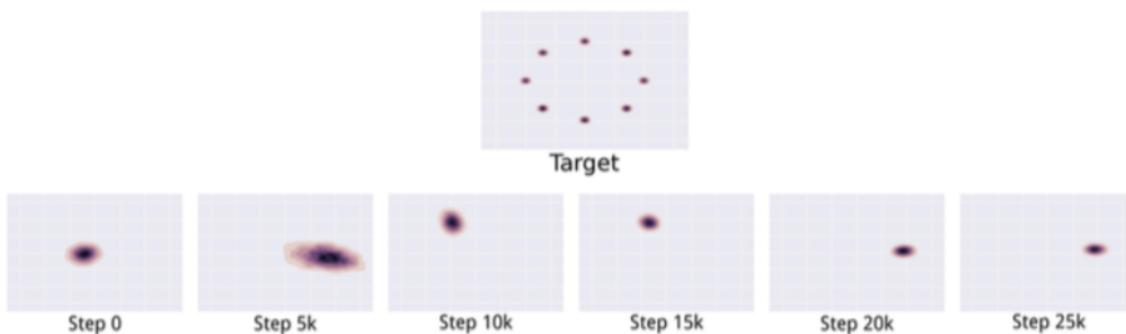
Nếu đi đủ sâu, ta sẽ nhận thấy rằng, việc GAN làm thực sự không phải là giúp cho phân phối tập ảnh giống hoàn toàn với phân phối thật. Mà phức tạp hơn, GAN đang cực tiểu hóa khoảng cách giữa hai phân phối. Điều này giúp GAN sinh ra được những thứ không chỉ giống mà còn là những điều tiềm ẩn ở trong dữ liệu có sẵn [12].

4.1.5 Mạng GAN không hoàn hảo

Một điều tệ khó tránh khỏi giống như nhiều mạng học sâu khác, GAN đôi khi cũng sẽ không hội tụ [3]. Xét một ví dụ trong không gian 2D (hình 4.1.6), mạng GAN chỉ có thể sinh ra phân phối liên thông tại xung qua 1 vị trí nào đó. Tuy nhiên, phân phối ta mong muốn lại không

⁶GMN không phải là một mô hình cụ thể mà là một loại phương pháp.

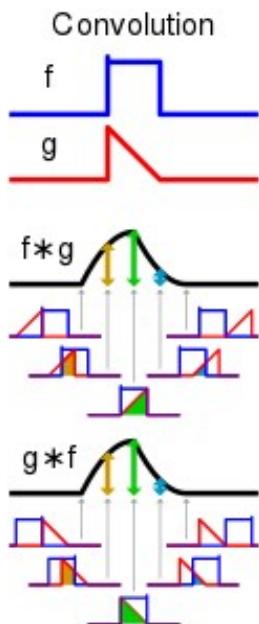
liên thông mà rải rác ở nhiều vùng, bộ phân biệt cũng vì thế sẽ cho điểm thấp (tức đánh giá không thật) do bộ sinh chưa đáp ứng đạt yêu cầu. Dựa vào đó, bộ sinh chuyển phân phối của mình sang chỗ khác. Và đương nhiên khi chuyển, là chuyển toàn bộ phân phối đi chứ không giữ lại một phần ở phân phối cũ. Nhưng trắc trở thay, tại một thời điểm, phân phối tập ảnh chỉ có thể ở một nơi trong rất nhiều nơi trong phân phối dữ liệu thật. Nên kết quả luôn luôn bị bộ phân biệt cho điểm thấp. Dẫn đến bộ sinh mãi mãi không bao giờ thấy mình có tiến bộ và phân kỳ.



Hình 4.1.6: GAN không có khả năng sinh ra được phân phối (hàng dưới) gần giống với phân phối được chọn (hàng trên) [37].

4.2 Mạng tích chập Unet

4.2.1 Khái quát về tích chập và mạng tích chập



Hình 4.2.1: Minh họa tích chập giữa hai hàm số f và g .

Sự đột phát trong ngành thị giác máy tính, hẳn khó tìm được ai trong những người đã có kinh nghiệm trong lĩnh vực này, phủ nhận việc công lớn thuộc về mạng tích chập - một loại mạng học sâu. Nó giúp giải quyết những bài toán phân loại hình ảnh, xem một tấm ảnh bất kì là ảnh về một con chó hay là của một con mèo, hoặc là một thứ gì khác [66]. Công nghệ nhận diện khuôn mặt được chính xác hơn cũng là nhờ ứng dụng mạng tích chập [28]. Nếu chỉ dừng lại ở mỗi khuôn mặt, thì cũng đã là rất thành công. Nhưng ý tưởng tích chập này còn có thể áp dụng được cho mọi loại đối tượng khác trong bài toán nhận diện [36] cũng như nhiều bài toán khác về xử lý ảnh [4].

Tích chập [71], là một phép toán giữa hai hàm số f, g dùng để tạo ra một hàm số khác $f * g$, hàm số này miêu tả hình thù của đồ thị hàm f bị thay đổi bởi hàm g (hình 4.2.1). Ta cũng có thể nói ngược lại là g bị thay đổi bởi f vì tích chập có tính chất giao hoán. Ý nghĩa của việc này, $f * g$ cho ta biết độ giống nhau giữa hai hàm f và g ở mỗi điểm.

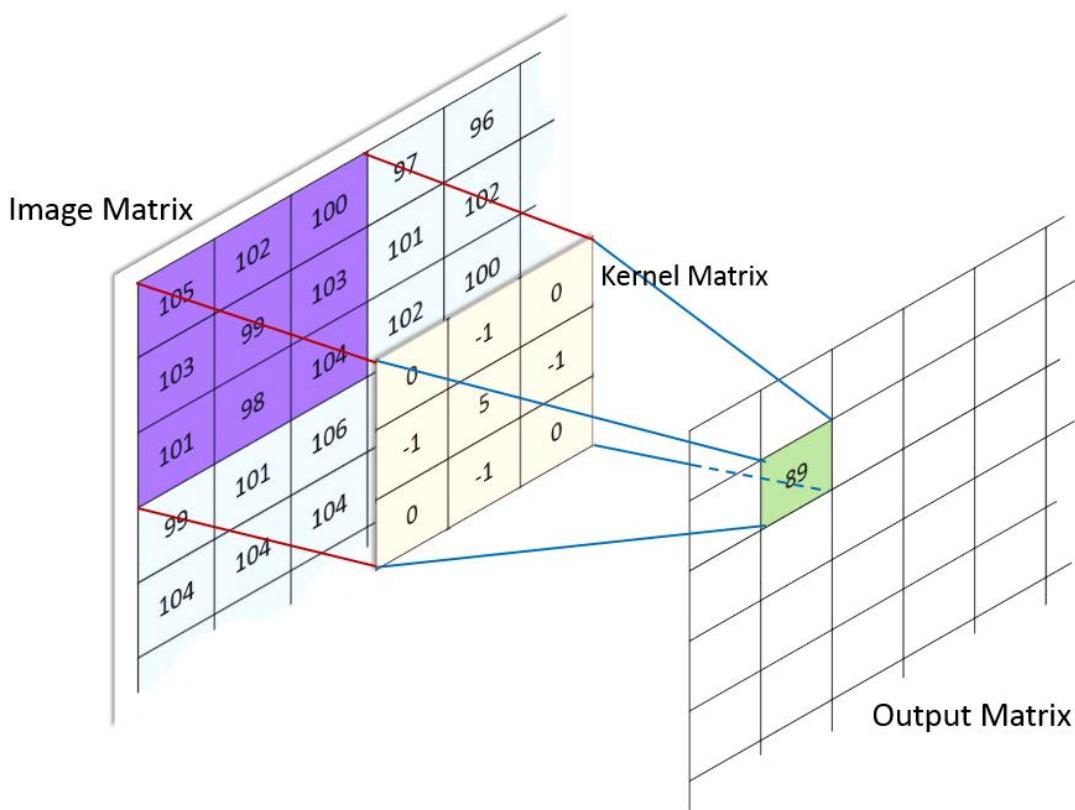
Ứng dụng điều này, các nhà khoa học đã thử lấy ảnh - là một ma trận số [2.1] $\mathcal{I} \in \mathbf{G}^{W \times H}$ làm f . Và sử dụng những ảnh bé hơn $\mathcal{K} \in \mathbb{R}^{\text{dim}(G) \times K \times K}$ miêu tả những cạnh, màu để làm g . Những ảnh này thường được gọi là các bộ lọc và mỗi lần tích chập chỉ dùng một bộ lọc. Để tính tích chập [77], họ trượt bộ lọc này đi khắp ảnh theo một bước nhảy S cố định, kèm thêm đệm P [86] cho ảnh để phù hợp kích thước nếu cần thiết. Ở từng vị trí được trượt, ta lần lượt trích xuất vùng ảnh tương ứng mà bộ lọc được

đặt lên $\mathcal{I}_{m:m+K-1,n:n+K-1} \in \mathbf{G}^{K \times K}$, sau đó sử dụng công thức:

$$(\mathcal{I} * \mathcal{K})_{m*,n*} = \sum_{k=1}^{\dim(\mathbf{G})} \sum_{i=0}^K \sum_{j=0}^K \mathcal{I}_{m+i,n+j}^{(k)} \mathcal{K}_{i+1,j+1}^{(k)} \quad (4.8)$$

Trong đó, tọa độ $(m*, n*)$ là vị trí kết quả của việc áp ảnh bộ lọc lên phần $\mathcal{I}_{m:m+K-1, n:n+K-1}$ (hình 4.2.2). Tọa độ này sẽ thay đổi tùy thuộc vào bước nhảy, đệm, cũng như là kích thước bộ lọc mà ta sử dụng. Kết quả sẽ là một số vô hướng, biểu thị sự giống nhau giữa bộ lọc và vùng ảnh được chọn.

Để triển khai công thức (4.8) trên máy tính, ta sẽ phải áp dụng những vòng lặp. Nhưng điều đó sẽ làm chậm tốc độ tính toán đi rất nhiều. Thay vào đó, một giải pháp được đưa ra đó là trải dài 2 ảnh ra thành một vector và tính tích vô hướng [74]. Việc trải ảnh ra để tính tích vô hướng là để tận dụng phép toán ma trận, tối ưu các luồng nhân để tăng tốc độ tính toán [11].



Hình 4.2.2: Minh họa tính tích chập với bộ lọc kích thước 3×3 .

Kích thước cuối cùng của ảnh đầu ra (giả sử ảnh đầu vào là một ảnh vuông, tức chiều rộng bằng chiều cao $W = H$) sau khi trượt một bộ lọc kích thước $K \times K$, bước nhảy S và đệm P từ ảnh gốc là [5]:

$$W_{\text{dầu ra}} = \left\lceil \frac{W_{\text{dầu vào}} - K + 2P}{S} \right\rceil + 1 \quad (4.9)$$

Công thức (4.9) sẽ là cần thiết khi ta quan tâm đến vùng nhận thức, để hiểu rõ về kiến trúc PatchGAN [4.3.2].

Mạng tích chập là mạng kết hợp nhiều bộ lọc theo một kiến trúc nào đó, nhằm trích lọc được các đặc trưng từ ảnh (như mắt, mũi, miệng từ ảnh mặt người). Điều này giúp cho bài

toán được giải quyết một cách đơn giản, hiệu quả hơn cho các tác vụ về ảnh, khi ta chỉ cần tập trung vào thứ cần tập trung nhờ bộ lọc. Thay vì những mạng học sâu thông thường, ta sẽ đưa tất cả những thông tin của ảnh đầu vào (ở mức đơn giản nhất, là toàn bộ điểm ảnh).

Khái niệm bộ lọc đã có trước khi có mạng học sâu tích chập, những bộ lọc này được con người tính toán thông qua những nghiên cứu. Rồi từ đó cũng trích xuất các đặc trưng và đưa vào các thuật toán học máy. Nhưng với học sâu, quá trình tìm ra bộ lọc được tích hợp vào trong quá trình huấn luyện mạng (hình 4.2.3). Giúp tìm ra những bộ lọc thích hợp nhất với mỗi tập dữ liệu bất kỳ.

Chỉ với phép tính tích vô hướng đơn giản như vậy, nhưng lại ảnh hưởng sâu sắc đến không chỉ trong mảng thị giác, mà còn được ứng dụng sang nhiều mảng khác. Khi người ta nhận thấy sức mạnh của tích chập, rất nhiều kiểu dữ liệu đã được cố gắng đẩy về dạng hình ảnh để ứng dụng mạng tích chập, chẳng hạn như âm thanh [26, 40].

Machine Learning



Deep Learning



Hình 4.2.3: Sự khác biệt giữa hai quá trình học máy và học sâu trong việc xử lý ảnh.

Trích lọc đặc trưng không phải là mục tiêu duy nhất khiến mọi người sử dụng tích chập. Mà còn là vì nó có thể hỗ trợ trong việc giảm các loại nhiễu cơ bản của ảnh đầu vào [13] (hình 4.2.4), làm mờ ảnh nếu cần thiết [59],...



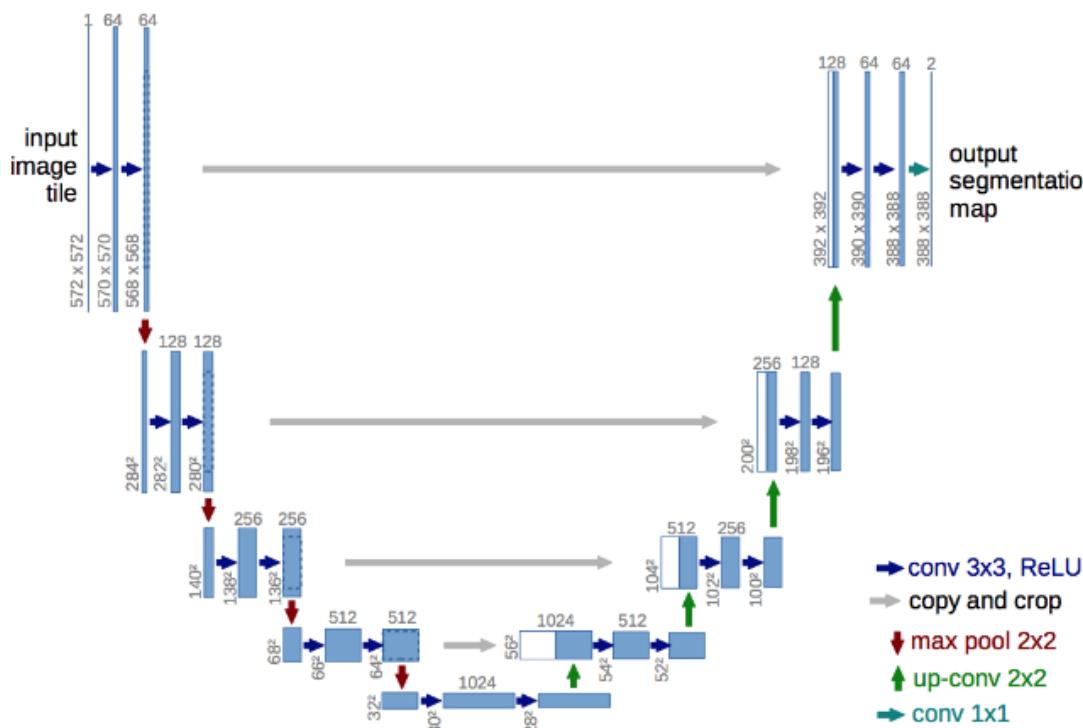
Hình 4.2.4: (a) Ảnh có nhiễu. (b) Ảnh sau khi được giảm nhiễu bằng bộ lọc 3×3 . (c) Ảnh sau khi được giảm nhiễu bằng bộ lọc 7×7 [62].

4.2.2 Kiến trúc Unet

Unet là một kiến trúc tích chập, được phát triển nhằm phân vùng các cấu trúc nơ-ron thần kinh trong não người. Kiến trúc này lần đầu được áp dụng đã giành chiến thắng trong cuộc thi EM segmentation challenge at ISBI 2012.⁷

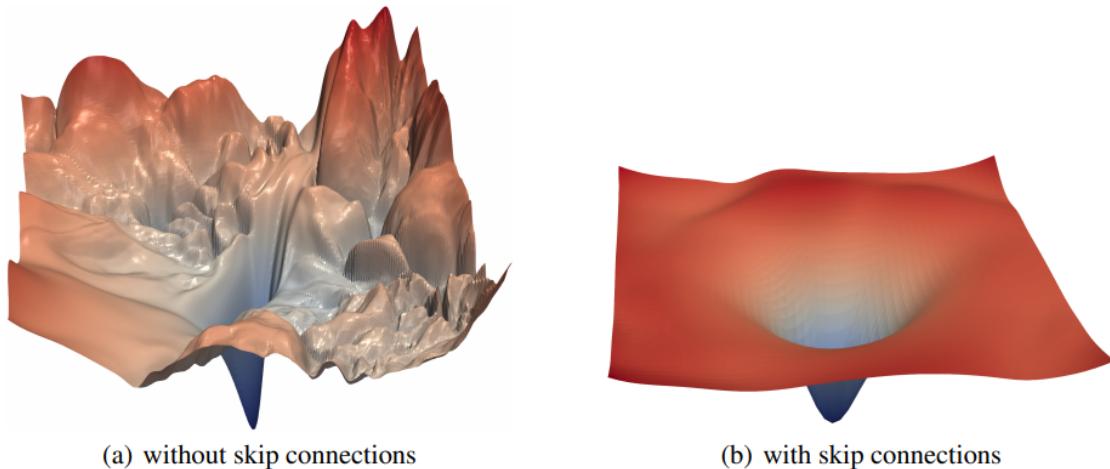
Unet bao gồm 2 nhánh đối xứng nhau hình chữ U nên được gọi là Unet. Nhánh ở bên trái là thu hẹp, còn phần mở rộng là nhánh ở bên phải. Mỗi phần sẽ thực hiện một nhiệm vụ riêng như sau:

- Phần thu hẹp: làm nhiệm vụ trích lọc đặc trưng để tìm ra bối cảnh của hình ảnh, tức để biết **CÁI GÌ**. Vai trò của phần thu hẹp tương tự như một bộ mã hóa. Một mạng tích chập sâu sẽ đảm nhận trích lọc đặc trưng. Lý do nhánh được gọi là thu hẹp vì kích thước dài và rộng, chính là độ phân giải, của các tầng giảm dần. Độ phân giải được giảm bằng cách sử dụng gộp cực đại [79], hoặc dùng tích chập có đệm và sải bước hợp lí.
- Phần mở rộng: Gồm các tầng đối xứng tương ứng với các tầng của nhánh thu hẹp có vai trò như một bộ giải mã. Ở phần này, độ phân giải của các tầng được tăng lại để biết được vị trí, tức **Ở ĐÂU**. Từ đó đánh dấu nhãn của từng điểm ảnh. Để làm được điều này, ta cần sử dụng kĩ thuật giải chập. Có nhiều phương pháp để giải chập, chẳng hạn như sao chép các giá trị điểm ảnh liền kề theo các kích thước cửa sổ [29]. Hoặc một phương thức khác là sử dụng tích chập giãn nở [85]. Nhưng được áp dụng nhiều nhất là tích chập chuyển vị [39] cùng với đệm và sải bước thích hợp.



Đặc trưng riêng trong cấu trúc của Unet đó là có những kết nối tắt đối xứng giữa các tầng của nhánh bên trái tương ứng với các tầng bên nhánh bên phải. Việc kết nối tắt theo kiến trúc Unet là một điểm mới so với kiến trúc mã hoá và giải mã thông thường nhằm bổ sung thêm thông tin và tăng độ chính xác. Một số mạng tích chập hiện đại sử dụng phương pháp trên [17, 18] để có thể giúp cho mô hình được sâu hơn, chính xác hơn.

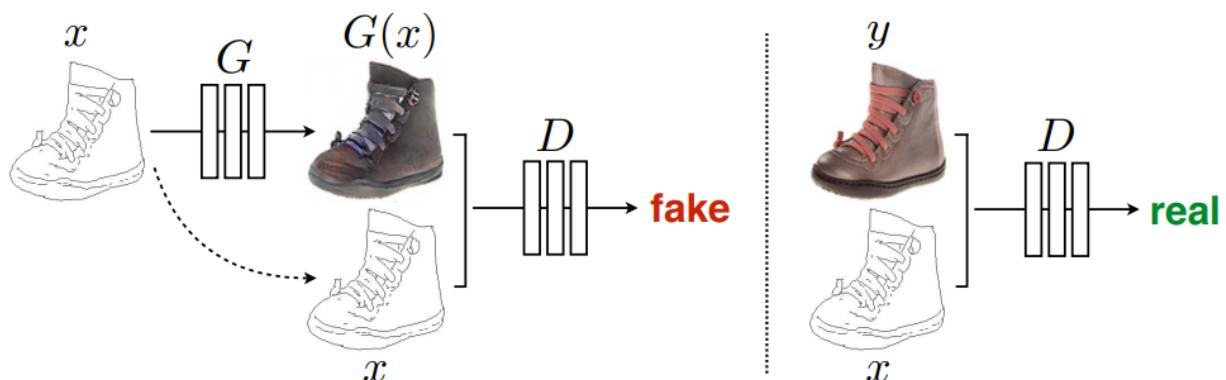
Ngoài ra kết nối tắt còn khắc phục hiện tượng gradient biến mất [1]. Một số nghiên cứu còn có thể chứng minh lý thuyết rằng, những mạng học sâu sử dụng kết nối tắt, ngoài hổ cực tiêu toàn cục, các hổ cực tiêu còn lại thường rất nồng [64].



Hình 4.2.6: Sự khác biệt của bề mặt hàm mục tiêu khi không (a) và có (b) sử dụng kết nối tắt [33].

4.3 Kiến trúc pix2pix

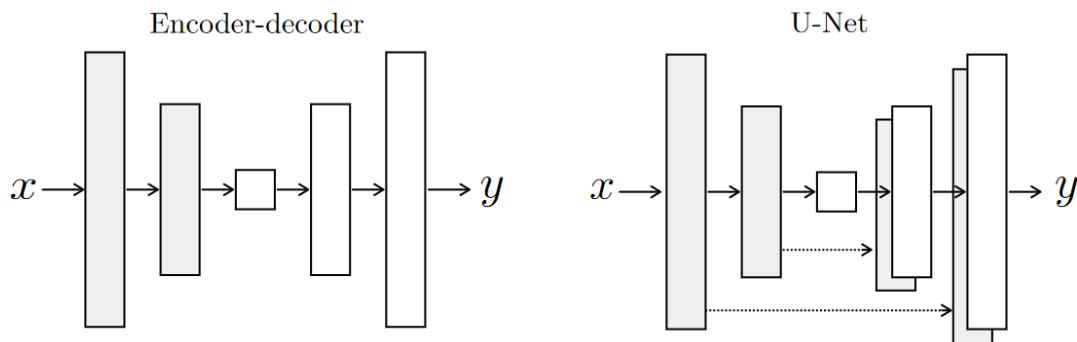
Bản thân kiến trúc GAN trong pix2pix cũng không khác mạng GAN [4.1] là bao. Gồm 2 mạng chính: bộ phân biệt và bộ sinh (hình 4.3.1). Tuy nhiên, có nhiều điều khác biệt giữa pix2pix so với GAN thông thường.



Hình 4.3.1: Mô hình pix2pix thêm màu sắc cho các nét vẽ để tạo ảnh màu. Một bài toán tương tự với bài toán tạo màu cho ảnh [22].

4.3.1 Khái quát mạng GAN trong pix2pix

Về bộ sinh, đầu vào bây giờ không còn là một vector nhiễu từ một phân phối chọn trước. Mà là một bức ảnh nguồn x . Sau đó bức ảnh được truyền vào một kiến trúc mạng tích chập để trích lọc đặc trưng và dùng đặc trưng này biến đổi lại thành ảnh đích $G(x)$. Các tác giả đã thử nghiệm 2 kiến trúc tích chập để xử lý việc này là Unet [4.2] và mã hoá-giải mã - một kiến trúc giống Unet nhưng không sử dụng kết nối tắt (hình 4.3.2). Giữa hai kiến trúc, Unet có kết quả khả quan hơn.



Hình 4.3.2: Tổng quát kiến trúc mạng mã hoá-giải mã và mạng Unet [22].

Bộ phân biệt giờ đây sẽ nhận vào một cặp ảnh. Trong đó luôn có ảnh x , được coi là điều kiện cho bộ phân biệt. Còn ảnh còn lại là ảnh đầu ra. Ảnh đầu ra có thể là ảnh được tạo ra từ bộ sinh $G(x)$, hoặc là nhãn của ảnh đầu vào được lấy từ tập dữ liệu huấn luyện y . Các cặp ảnh sẽ được gán nhãn là thật nếu là (x, y) . Ngược lại sẽ là giả nếu đi kèm là $(x, G(x))$.

4.3.2 Bộ phân biệt Patch trong pix2pix

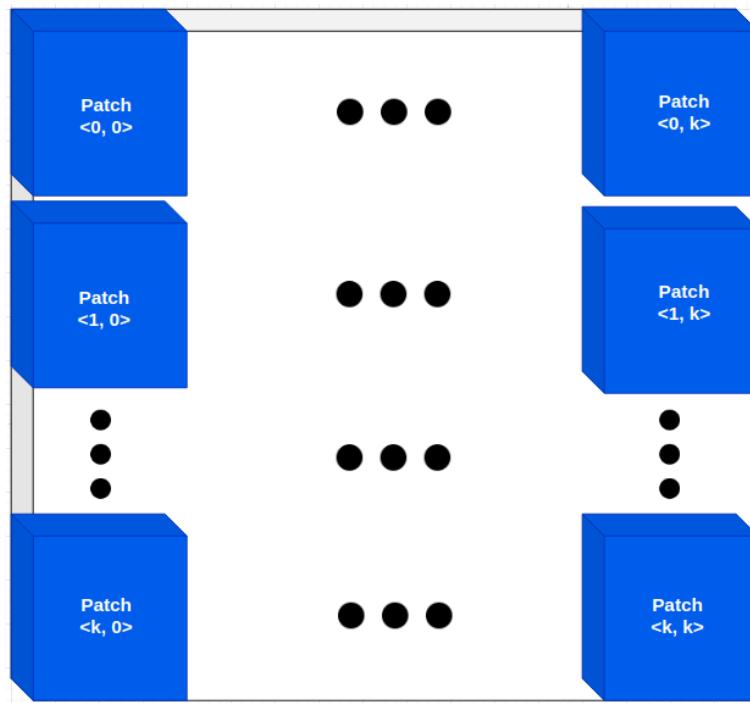
Một điểm khác biệt nữa của bộ phân biệt trong mô hình pix2pix so với những bộ phân biệt trong các mô hình GAN thông thường, là bộ phân biệt của chúng ta là làm việc theo Patch. Một bộ phân biệt thông thường chỉ trả về một số vô hướng thì bộ phân biệt Patch sẽ trả về một ma trận $\mathbf{P}_D \in \mathbb{R}^{k \times k}$ (hình 4.3.3). Mỗi phần tử của ma trận $[\mathbf{P}_D]_{ij}$ là kết quả phân loại một vùng $N \times N$ của đầu vào là thật hay giả. Những vùng $N \times N$ này được gọi là vùng nhận thức (hình 4.3.4).

Về bản chất, bộ phân biệt Patch vẫn là một kiến trúc mạng tích chập gồm nhiều tầng tích chập liên tiếp nhau. Nhưng chúng ta không thực hiện làm phẳng ở gần cuối để truyền qua các lớp kết nối đầy đủ rồi cho ra duy nhất một vô hướng. Mà thay vào đó, tính toán để ra được dự báo xác suất trên mỗi Patch rồi vào ảnh thật. Xác suất để toàn bộ đầu vào là thật sẽ là trung bình cộng của toàn bộ k^2 Patch. Cách tính như vậy sẽ mang lại hiệu quả nếu áp dụng trên các Patch có kích thước lớn vì có vùng nhận thức lớn hơn. Kết quả xác suất trung bình của nhiều Patch cũng sẽ chuẩn xác hơn.

Sau khi đã thử qua một số kích thước Patch như $1 \times 1, 16 \times 16, 70 \times 70, 286 \times 286$ thì kích thước Patch tốt nhất theo [22] là 70×70 . Kiến trúc các tầng của một bộ phân biệt Patch 70×70 có thể biểu diễn đơn giản như sau:

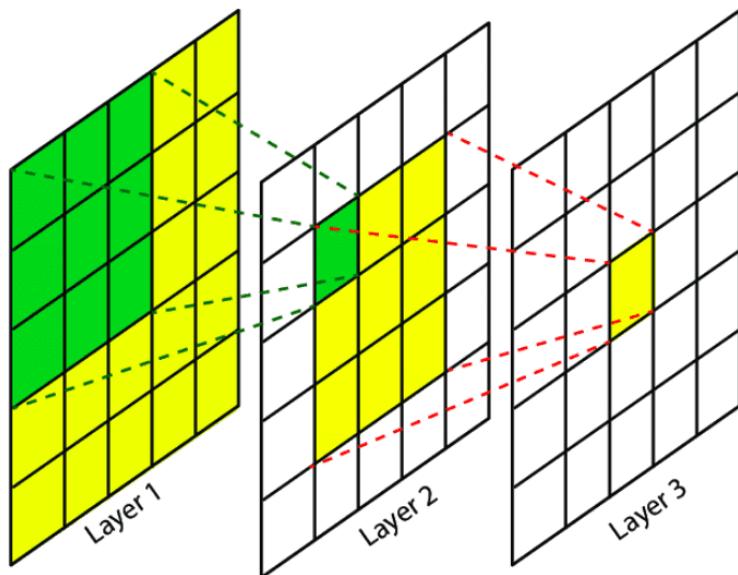
$$I \rightarrow 2C64 \rightarrow 2C128 \rightarrow 2C256 \rightarrow 1C512 \rightarrow O \quad (4.10)$$

Với sCk là một tầng tích chập gồm k bộ lọc và có sải bước là s . Tầng cuối cùng O sử dụng sải bước là 1 và chỉ sử dụng duy nhất 1 bộ lọc để trả ra ma trận \mathbf{P}_D , rồi kích hoạt hàm sigmoid



Hình 4.3.3: Hình ảnh giả sử được đầu vào chia thành $k \times k$ Patch [30].

để lấy xác suất. Tất cả các tầng đều sử dụng kích thước bộ lọc là 4×4 cùng với kích thước đệm là 1.



Hình 4.3.4: Mô tả vùng nhận thức qua các tầng.

Ta hoàn toàn có thể tính ngược lại được vùng nhận thức dựa vào thông tin kích thước bộ lọc và sải bước. Từ (4.9), bằng các phép biến đổi đại số cơ bản, ta suy ra được:

$$W_{\text{đầu vào}} = K - 2P + S (W_{\text{đầu ra}} - 1) \quad (4.11)$$

Tuy nhiên, khi suy ngược lại để tính vùng nhận thức, vùng nhận thức ta cần quan tâm vùng ảnh không có đệm. Do đó, điều chỉnh công thức (4.11), kích thước vùng nhận thức F

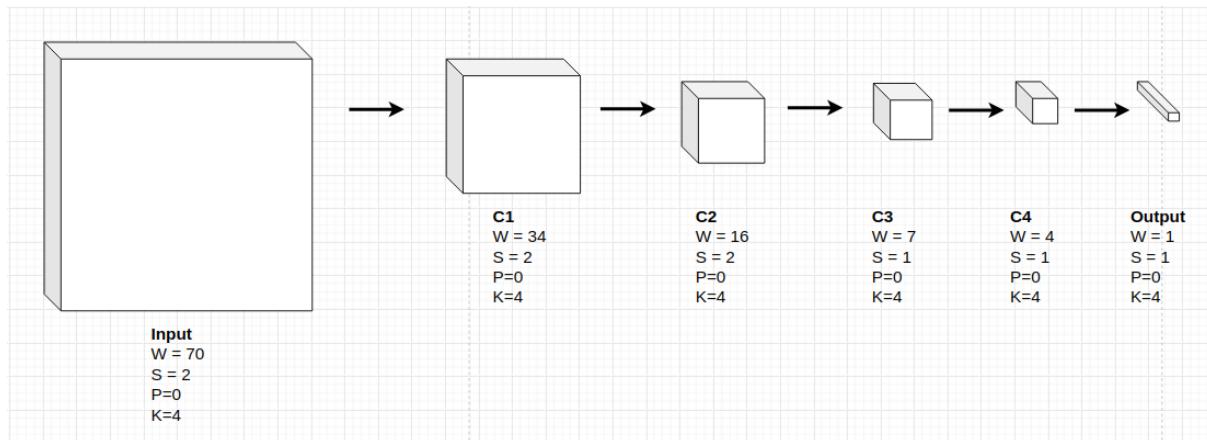
từng tầng sẽ là:

$$F_L = K + S(F_{L+1} - 1) \quad (4.12)$$

Áp dụng công thức (4.12), ta sẽ tính được kích thước vùng nhận thức của mỗi giá trị đầu ra của bộ phân biệt Patch 70×70 (4.10) theo như kết quả sau đây:

$$\begin{aligned} F_4 &= K + S_4(F_5 - 1) = 4 + 1(1 - 1) = 4 \\ F_3 &= K + S_3(F_4 - 1) = 4 + 1(4 - 1) = 7 \\ F_2 &= K + S_2(F_3 - 1) = 4 + 2(7 - 1) = 16 \\ F_1 &= K + S_1(F_2 - 1) = 4 + 2(16 - 1) = 34 \\ F_0 &= K + S_0(F_1 - 1) = 4 + 2(34 - 1) = 70 \end{aligned}$$

Ta có thể mô tả tổng quát kiến trúc các lớp mạng tích chập áp dụng trên một Patch 70×70 (hình 4.3.5):



Hình 4.3.5: Sơ đồ tổng quát kiến trúc các lớp mạng tích chập áp dụng trên một Patch 70×70 .

4.3.3 Hàm mục tiêu (mắt mát) pix2pix

Hàm mục tiêu của mạng GAN có điều kiện pix2pix gần giống với (4.5), được biểu diễn như sau:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y} [\log D(x, y)] + \mathbb{E}_{x,z} [\log (1 - D(x, G(x, z)))] \quad (4.13)$$

Trước khi giải thích ý nghĩa của công thức, có một điều chưa được rõ ràng trong (4.13), là một mâu thuẫn nhỏ ở $\log D(x, y)$ và $\log (1 - D(x, G(x, z)))$. Như đã biết, hàm $\log(\cdot)$ là một hàm nhận vào một số vô hướng, nhưng theo [4.3.2] trình bày, bộ phân biệt của kiến trúc pix2pix sẽ trả ra một ma trận. Cũng ở trong đã trình bày [4.3.2]: “Xác suất để toàn bộ đầu vào là thật sẽ là trung bình cộng của toàn bộ k^2 Patch”. Tức là, ta có thể hiểu ngầm giá trị bộ phân biệt Patch $D(\cdot)$ thành một số vô hướng như sau:

$$D(\cdot) = \frac{1}{k^2} \sum_{i=1}^k \sum_{j=1}^k (\mathbf{P}_D)_{ij} \quad (4.14)$$

Tuy nhiên, nếu đem cách hiểu (4.14) đưa vào trong các hàm mất mát để tối ưu, ta sẽ có xu hướng làm trung bình của các phần tử trong \mathbf{P}_D gần với 0 hoặc 1, thay vì cái ta thực sự muốn chính là mỗi phần tử $(\mathbf{P}_D)_{ij}$ gần với 0 hoặc 1. Vậy nên, giá trị $\log(f(D))$ trong những hàm mất mát nên được hiểu ngầm là:

$$\log(f(D)) = \frac{1}{k^2} \sum_{i=1}^k \sum_{j=1}^k \log(f(\mathbf{P}_{D_{ij}})) \quad (4.15)$$

Quay trở lại với (4.13). Như đã đề cập [4.3.1], bộ phân biệt bây giờ sẽ nhận một cặp (x, y) với x là điều kiện thay vì chỉ mỗi y như GAN thông thường. Việc thử nghiệm mức độ quan trọng của điều kiện x đã được thử nghiệm [22] bằng cách sử dụng hàm mất mát bỏ đi x trong đầu vào bộ phân biệt:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_y [\log D(y)] + \mathbb{E}_{x,z} [\log(1 - D(G(x, z)))] \quad (4.16)$$

Và kết quả thử nghiệm này cho thấy rằng bộ phân biệt chỉ cần đầu ra giống thật chứ không quan tâm đầu ra đó là đầu ra cho đầu vào thế nào. Sau khi xem xét các kết quả thử nghiệm, họ nhận thấy rằng bộ sinh rời vào tình trạng tạo ra những ảnh đích gần giống nhau bất chấp ảnh đầu vào. Hiển nhiên sẽ dẫn đến một kết quả không tốt.

Không những phải lừa bộ phân biệt, bộ sinh còn phải làm sao tạo ra kết quả giống với nhãn. Chính vì điều này, một số nghiên cứu cũng chỉ ra rằng, kết hợp với một hàm mất mát thông thường sẽ cho kết quả tốt hơn, chẳng hạn như chuẩn 2 [46]. Trong khung mô hình pix2pix, chuẩn 1 được ưu tiên hơn so với chuẩn 2 vì chuẩn 2 cho kết quả bị mờ.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z} [\|y - G(x, z)\|_1] \quad (4.17)$$

Lý do để chọn chuẩn bậc 1 thay vì chuẩn bậc 2 có thể được lý giải một cách trực quan, là do chuẩn bậc 2 có xu hướng trừng phạt mất mát tiêu cực hơn so với chuẩn bậc 1 [35]. Điều này làm bó buộc kết quả dự đoán của bộ sinh khiến cho mô hình có xu thế bảo thủ, sẽ đưa dự đoán bằng cách lấy giá trị trung bình để tối thiểu sự trừng phạt. Thay vì vậy, chuẩn bậc 1 lại nhẹ nhàng trong việc trừng phạt hơn, khiến cho mô hình có thể sáng tạo để đưa ra những dự đoán phiêu lưu hơn.

Từ (4.13) và (4.17), hàm mục tiêu của pix2pix là:

$$G^* = \arg \min_G \max_D \{\mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)\} \quad (4.18)$$

Hệ số λ là hệ số cân bằng, để giúp cho sự chênh lệch giữa hai hàm mất mát không bị áp đảo nghiêng về một bên, khiến một hàm bị lu mờ trong quá trình huấn luyện. Thông thường giá trị hàm mất mát của GAN sẽ có giá trị lớn hơn nhiều so với hàm chuẩn 1. Nên xu hướng chọn λ sẽ là một số lớn hơn 1.

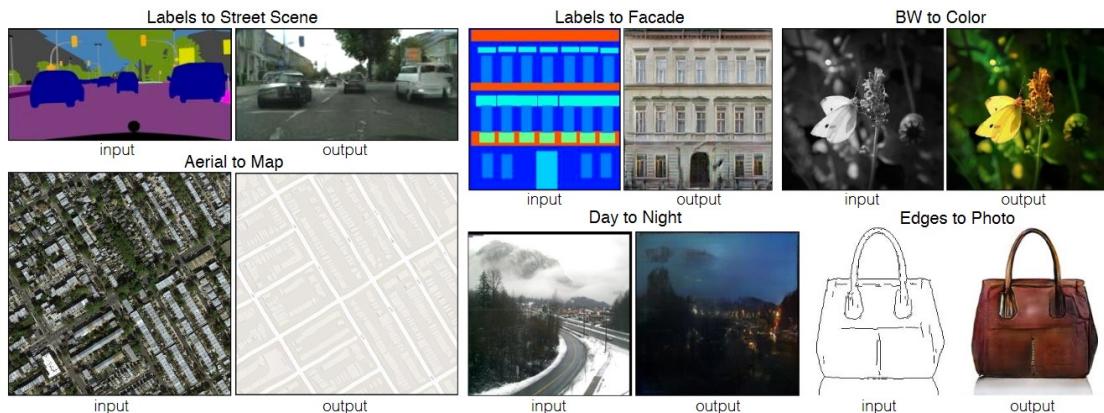
Một điều nữa cần được đề cập, chính là phần nhiễu z . Nhìn chung, nếu không có nhiễu này, mô hình vẫn có thể học được một cách bình thường. Theo như tác giả cho biết [22], hạn chế việc này là kết quả phân phối của bộ sinh sẽ khó có thể phù hợp được với những phân phối lâ. Nhưng nếu đưa nhiễu vào như cách truyền thống của mạng GAN bình thường sử dụng, bộ sinh sẽ học được cách để ngó lơ những nhiễu này. Nhóm tác giả đã thay đổi chiến lược này bằng cách tạo nhiễu thông qua kỹ thuật dropout [58] trong cả giai đoạn huấn luyện và cả thử nghiệm.

Ở một phương diện khác, mô hình không thực sự cần đến nhiễu nếu đầu vào đủ phức tạp [48]. Lúc đó đầu vào đóng vài trò như nhiễu. Chẳng hạn với bài toán tô màu trong đồ án này, đầu vào là một bức ảnh xám được đánh giá là đủ phức tạp, nên thành phần nhiễu được phép bỏ qua.

Chương 5

Kiến trúc mô hình cho bài toán tạo màu dựa vào khung mô hình pix2pix

Khung mô hình pix2pix được tạo ra để cho các vấn đề chuyên biệt về chuyển đổi ảnh-ảnh.



Hình 5.0.1: Các ứng dụng của khung mô hình pix2pix [22].

Vô số các ứng dụng liên quan đã được tạo ra (hình 5.0.1), tiêu biểu như:

- Chuyển đổi trời tối sang trời sáng.
- Tái tạo phong cách hội họa của những họa sĩ nổi tiếng đã qua đời như Van Gogh, Picasso, Monet,... dựa trên những bức tranh họ để lại.
- Chuyển đổi từ ảnh được phân vùng sang ảnh thật.
- Thêm màu sắc cho các nét vẽ để tạo thành ảnh màu hoàn chỉnh.
- Tạo màu cho ảnh xám.
- Hay một ứng dụng tuyệt vời nhất mà chúng ta sử dụng hằng ngày, đó chính là chuyển đổi bức ảnh chụp từ vệ sinh sang Google Maps.

Đồ án này triển khai một mô hình dựa theo khung mô hình pix2pix để giải quyết bài toán tạo màu cho ảnh xám. Một ứng dụng của khung mô hình trên. Kích thước ảnh được chọn để xử lý sẽ là 256×256 , một kích thước phổ biến trong những mạng tích chập hiện nay và phù hợp cho mô hình được triển khai.

5.1 Phát biểu bài toán, cách áp dụng vào khung mô hình pix2pix và hàm mất mát

Bài toán tạo màu có thể được phát biểu như sau:

Cho một ảnh xám đầu vào $\mathcal{I}_{\text{đầu vào}} \in \mathbf{G}_{\mathbf{L}^*}^{256 \times 256}$, ta cần xây dựng một ảnh xạ:

$$\mathcal{G} : \mathbf{G}_{\mathbf{L}^*}^{256 \times 256} \rightarrow (\mathbf{G}_{\mathbf{a}^*} \times \mathbf{G}_{\mathbf{b}^*})^{256 \times 256} \quad (5.1)$$

$$\widehat{\mathcal{I}} = (\mathcal{I}_{\text{đầu vào}}, \mathcal{G}(\mathcal{I}_{\text{đầu vào}})) \in (\mathbf{G}_{\mathbf{L}^* \mathbf{a}^* \mathbf{b}^*})^{256 \times 256} \quad (5.2)$$

Sao cho ảnh $\widehat{\mathcal{I}}$ (5.2) là một ảnh màu trong không gian màu $\mathbf{L}^* \mathbf{a}^* \mathbf{b}^*$ có được sự hợp lý, chân thực với mắt người.

Việc xây dựng ảnh xạ sẽ được dẫn đường bởi ảnh xạ:

$$\mathcal{D} : (\mathbf{G}_{\mathbf{L}^* \mathbf{a}^* \mathbf{b}^*})^{256 \times 256} \rightarrow \mathbb{R}^{70 \times 70} \quad (5.3)$$

Trong mạng GAN pix2pix được huấn luyện, bộ sinh G sẽ là ảnh xạ \mathcal{G} cần tìm. Còn ảnh xạ \mathcal{D} là bộ phân biệt D có nhiệm vụ phân biệt ảnh màu đưa vào là thật hay giả, hỗ trợ xây dựng ảnh xạ \mathcal{G} .

Bộ sinh theo kiến trúc mạng Unet với đầu vào là một ảnh $\mathcal{I}_{\text{đầu vào}}$ chỉ có một kênh biểu thị cho cường độ sáng mỗi điểm ảnh. Bức ảnh sẽ được truyền vào một kiến trúc mạng tích chập để trích lọc đặc trưng qua quá trình mã hóa. Ở quá trình này, kích thước đầu ra ở mỗi tầng sẽ giảm dần theo bội 2. Trong giai đoạn giải mã, các đặc trưng được kết hợp với phía trái nhánh chữ U là phần mã hóa, để biến đổi lại thành ảnh đích bằng tích chập chuyển vị. Và mỗi tầng sẽ có kích thước tăng dần cũng theo bội 2. Kết quả trả ra sau cùng là hai kênh màu $\widehat{\mathcal{I}}_{\text{đầu ra}} = \mathcal{G}(\mathcal{I}_{\text{đầu vào}})$. Khi đó, ảnh giả được bộ sinh tạo ra là $\widehat{\mathcal{I}} = (\mathcal{I}_{\text{đầu vào}}, \widehat{\mathcal{I}}_{\text{đầu ra}})$.

Đầu vào của bộ phân biệt có thể là ảnh $\mathcal{I} = (\mathcal{I}_{\text{đầu vào}}, \mathcal{I}_{\text{đầu ra}})$ với $\mathcal{I}_{\text{đầu ra}} \in \mathbf{G}_{\mathbf{a}^*} \times \mathbf{G}_{\mathbf{b}^*}$ là nhãn tương ứng với $\mathcal{I}_{\text{đầu vào}}$ trong tập dữ liệu huấn luyện. Với đầu vào này, ta sẽ gán nhãn cho là $\mathbf{P}_D = \mathbf{J}_{70 \times 70}$ vì là dữ liệu thật. Còn khi dữ liệu đầu vào là $\widehat{\mathcal{I}}$, ta sẽ có nhãn cho bộ phân biệt là $\mathbf{P}_D = \mathbf{O}_{70 \times 70}$.

Dựa vào (4.18) và bỏ nhiều [4.3.3], ta có hàm mục tiêu cho bài toán tô màu là:

$$\begin{aligned} \mathcal{G}^* = \arg \min_{\mathcal{G}} \max_{\mathcal{D}} & \left\{ \mathbb{E}_{\mathcal{I}} [\log \mathcal{D}(\mathcal{I})] + \mathbb{E}_{\mathcal{I}_{\text{đầu vào}}} [\log (1 - \mathcal{D}(\mathcal{I}_{\text{đầu vào}}, \mathcal{G}(\mathcal{I}_{\text{đầu vào}})))] \right. \\ & \left. + \lambda \underbrace{\mathbb{E}_{\mathcal{I}_{\text{đầu vào}}, \mathcal{I}_{\text{đầu ra}}} \|\mathcal{I}_{\text{đầu ra}} - \mathcal{G}(\mathcal{I}_{\text{đầu vào}})\|}_{\mathcal{L}_{L1}(\mathcal{G})} \right\} \end{aligned} \quad (5.4)$$

Hệ số λ được chọn giống như [22] gợi ý là $\lambda = 100$.

5.2 Kiến trúc bộ sinh pix2pix và bộ sinh học chuyển giao từ xương sống ResNet18

Trong mô hình tạo màu, được quan tâm hơn cả là bộ sinh vì nó là ánh xạ \mathcal{G} ta cần tìm để giải bài toán tạo màu. Kiến trúc tổng quát bộ sinh được đề xuất trong [22] như sau:

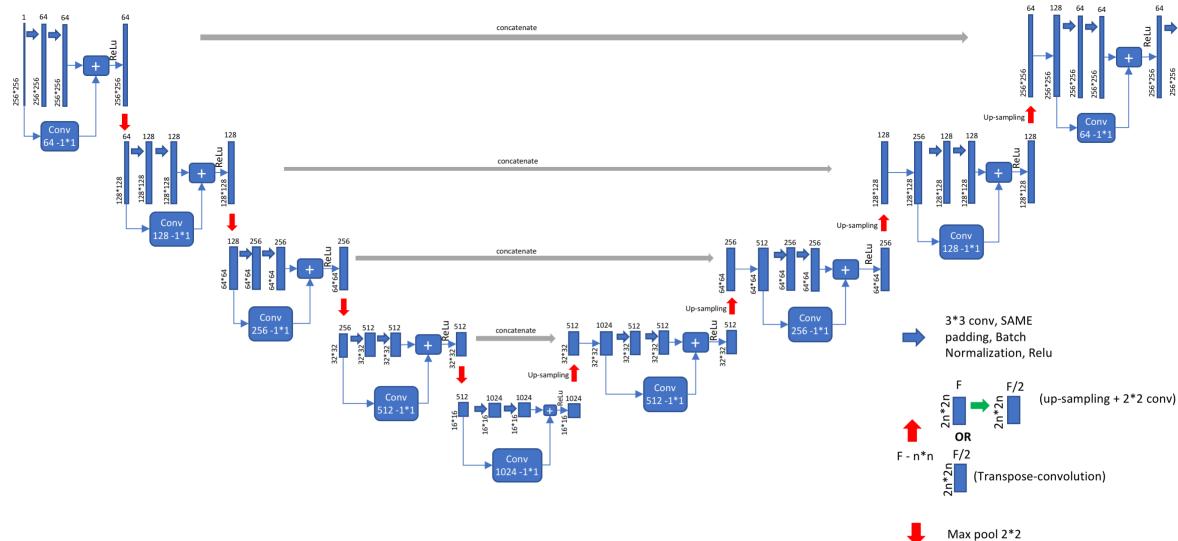
$$\text{nhánh trái: } I \rightarrow C64 \rightarrow C128 \rightarrow C256 \rightarrow C512 \rightarrow C512 \rightarrow C512 \rightarrow C512 \quad (5.5)$$

$$\text{nhánh phải: } \rightarrow D512 \rightarrow D1024 \rightarrow D1024 \rightarrow D1024 \rightarrow D512 \rightarrow D256 \rightarrow D128 \rightarrow D2 \quad (5.6)$$

Trong đó, C_k là một tầng tích chập k bộ lọc, D_k là một tầng giải chập bằng tích chập chuyển vị với k bộ lọc. Cả 2 quá trình chập và giải chập đều sử dụng kích thước bộ lọc là 4, sải bước 2 và đệm 1.

Xây dựng mô hình theo kiến trúc trên không quá khó. Nhưng huấn luyện để có được kết quả tốt tương tự, khi hạn chế về số lượng và chất lượng của tập dữ liệu, cũng như phần cứng máy tính bị giới là khá thử thách [7.1].

Cải thiện mô hình mà vẫn tiết kiệm chi phí, phương pháp học chuyển giao được áp dụng để làm điều này. Kiến trúc cho bộ sinh vẫn sẽ là kiến trúc theo mạng Unet, tuy nhiên sẽ được chuyển giao bằng cách lấy xương sống là ResNet18 [38]. Có 3 họ mạng được chủ yếu dùng để làm xương sống cho Unet là VGG, ResNet và Xception [88]. Trong những họ mạng trên, ResNet18 thuộc họ mạng ResNet có kích thước không quá lớn [27], phù hợp để huấn luyện với phần cứng bị giới hạn.



Hình 5.2.1: Một kiểu mạng Unet được khởi tạo theo phong cách ResNet [57].

Một lưu ý là toàn bộ phần tích chập của ResNet18 (đã bỏ đi các tầng kết nối dày đặc) sẽ chỉ làm ở phía mã hoá, tức nhánh bên trái của chữ U chứ không hề xây dựng lên toàn bộ cả mạng Unet. Phần mã hoá sẽ được tự động khởi tạo thích hợp bởi thư viện fastai¹.

¹fastai Documentation, “Dynamic UNet”, <https://docs.fast.ai/vision.models.unet.html>

Chương 6

Triển khai mô hình

6.1 Ngôn ngữ lập trình và thư viện chính

Ngôn ngữ lập trình được sử dụng để triển khai mô hình là **Python** với sự hỗ trợ chính của thư viện **PyTorch**. Để cài đặt những công cụ này, có thể theo đường dẫn bên dưới:

- Python: <https://www.python.org/downloads/>
- PyTorch <https://pytorch.org/get-started/locally/>

Thư viện **fastai** (bản mới nhất) cũng được cài đặt cho việc xây dựng mạng Unet và một số tác vụ liên quan. Để tải thư viện này về, mở cửa sổ dòng lệnh (đã được nhúng môi trường Python) rồi thực thi mã lệnh:

```
pip install fastai --upgrade
```

Toàn bộ mã nguồn, cũng như những tệp liên quan trong quá trình hoàn thành mô hình, có thể tìm thấy trong thư mục đồ án tại:

- Github: https://github.com/dee-ex/EE3151_SEM202_PROJECT

Để có thể biết được đầy đủ những thư viện cũng như phiên bản thư viện được sử dụng, tham khảo tệp tin **requirements.txt** trong đường dẫn thư mục đồ án.

6.2 Tập dữ liệu, chuẩn hóa và làm giàu dữ liệu

Tập dữ liệu được chọn sử dụng là tập **COCO**¹ (hình 6.2.1) có trong thư viện **fastai**, ta có thể dễ dàng tải về. Số lượng ảnh được chọn để huấn luyện là 10,000. Tất cả đều được chọn ngẫu nhiên và có xáo trộn.

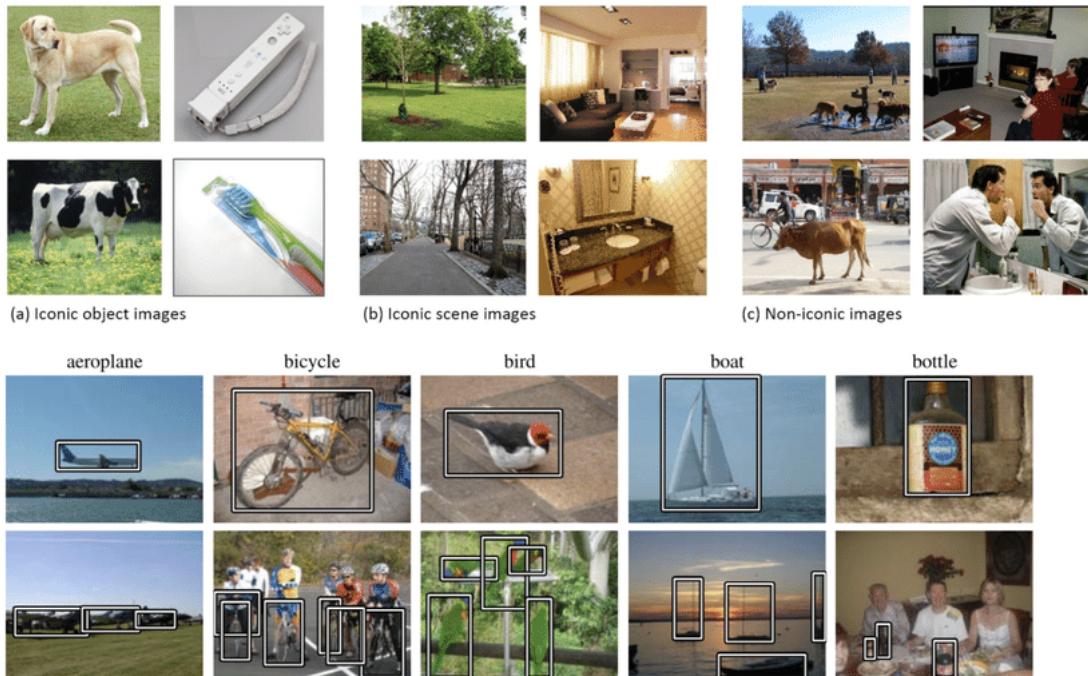
Chuẩn hóa dữ liệu đầu vào là bước tiền xử lí giúp tăng tốc độ hội tụ cho mô hình, giảm sự phụ thuộc của gradient vào tỉ lệ các tham số và một số lợi ích khác [8, 2]. Ở đây, 3 kênh dữ liệu sẽ được chuẩn hóa cùng về một khoảng $[-1, 1]$:

$$\mathbf{G}_{\text{chuẩn hóa } \mathbf{L}^*} = \frac{\mathbf{G}_{\mathbf{L}^*}}{50} - 1 \quad (6.1)$$

$$\mathbf{G}_{\text{chuẩn hóa } \mathbf{a}^*} = \frac{\mathbf{G}_{\mathbf{a}^*}}{110} \quad (6.2)$$

$$\mathbf{G}_{\text{chuẩn hóa } \mathbf{b}^*} = \frac{\mathbf{G}_{\mathbf{b}^*}}{110} \quad (6.3)$$

¹COCO Dataset - Download, <https://cocodataset.org/#download>



Hình 6.2.1: Hình ảnh trích từ tập dữ liệu COCO.

Vì $\mathbf{G}_{\mathbf{L}^*} \in [1, 100]$, nên ta chuẩn hoá như (6.1) là hết sức bình thường. Riêng với (6.2) và (6.3) lại có thể gây một chút khó hiểu. Bởi vì thư viện được sử dụng để chuyển đổi màu là **skimage**. Và trong **skimage** thì $\mathbf{G}_{\mathbf{a}^*} \in [-86, 98]$ và $\mathbf{G}_{\mathbf{b}^*} \in [-108, 94]$ [24]. Do đó, cận trên 110 để chuẩn hoá. Ta cũng hoàn toàn có thể chọn riêng cận trên để chuẩn hoá. Giả sử như của $\mathbf{G}_{\mathbf{a}^*}$ là 100, còn $\mathbf{G}_{\mathbf{b}^*}$ là 110. Dĩ nhiên, trong cả hai hướng chuẩn hoá trên thì một số trường hợp khi đưa từ giá trị chuẩn hoá về lại giá trị gốc, sẽ không còn khớp với thư viện **skimage**. Tuy nhiên, phần đó là thiểu số, không quá đáng kể.

Bên cạnh chuẩn hoá không gian màu, kích thước ảnh cũng sẽ được điều chỉnh lại thành ảnh vuông kích thước 256×256 nếu cần thiết. Sử dụng lật đối xứng theo trực tung (hình 6.2.2) để làm giàu dữ liệu cho việc huấn luyện. Như vậy, tổng số lượng dữ liệu mà mô hình được học sẽ là $10,000 \times 2 = 20,000$.

6.3 Huấn luyện mô hình

Cách thức huấn luyện cho mô hình pix2pix sử dụng Python-PyTorch được tham khảo chính từ [47, 56, 89]. Đồng thời cũng có thêm một vài tham khảo về việc sử dụng Keras [44, 30, 7] để huấn luyện những mô hình tương tự.

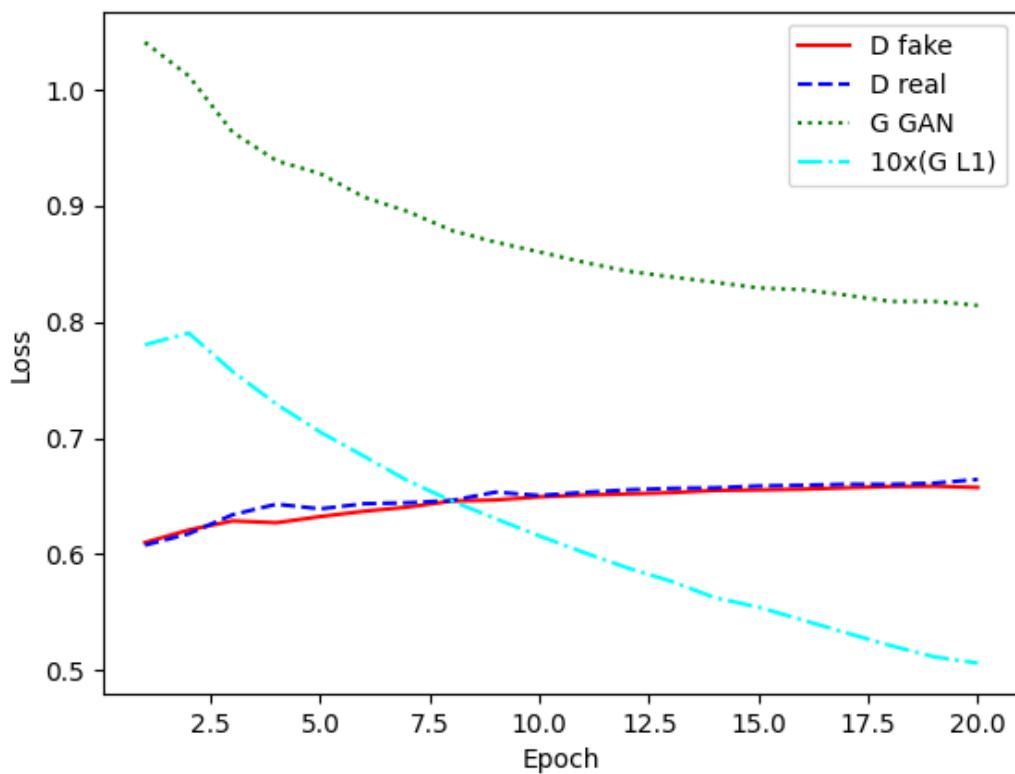
Vì mô hình bộ sinh phức tạp và quan trọng hơn so với bộ phân biệt, vậy nên để cho quá trình huấn luyện GAN được ổn định hơn, cũng như tránh bộ phân biệt hội tụ sớm, ta sẽ tiền huấn luyện [16] độc lập bộ sinh bằng $\mathcal{L}_{L1}(\mathcal{G})$ (5.4) qua 20 epoch với kích thước batch là 16. Mỗi epoch mất khoảng 6–7 phút khi huấn luyện trên Google Colab². Thuật toán tối ưu được sử dụng là Adam [31] với tốc độ học $\alpha = 10^{-4}$ và mô men $\beta_1 = 0.9, \beta_2 = 0.999$.

Tải các thông số của bộ sinh đã được tiền huấn luyện độc lập trước đó, đưa vào quá trình huấn luyện một mạng đối nghịch. Thuật toán tối ưu vẫn sẽ dùng là Adam với tốc độ học

²Google Colab, <https://colab.research.google.com/>



Hình 6.2.2: Minh họa việc lật đổi xứng theo trực tung.



Hình 6.3.1: Giá trị các thành phần mất mát của bộ phân biệt và bộ sinh qua từng epoch.

$\alpha = 2.10^{-4}$ và mô men $\beta_1 = 0.5, \beta_2 = 0.999$. Mô hình được huấn luyện qua 20 epoch với kích thước batch là 16. Các batch sẽ có số chiều là $16 \times 1 \times 256 \times 256$ (kích thước batch, số kênh, số hàng, số cột). Mỗi epoch mất khoảng 10–16 phút bằng việc sử dụng Google Colab để thực thi.

Nhận thấy xu hướng của giá trị mất mát khi huấn luyện mô hình GAN (hình 6.3.1) là: giá trị mất mát của bộ phân biệt (đường nét liền đỏ và đường nét đứt lam) tăng thì giá trị mất mát



của bộ sinh (đường nét chấm lục) sẽ giảm. Điều này khá dễ dàng để lý giải. Khi bộ phân biệt đạt đến một ngưỡng thông minh nhất định, bộ sinh thì lại càng thông minh qua mỗi epoch, hiển nhiên sẽ đánh lừa được bộ phân biệt. Chuyện này là hợp lí với nguyên lí non-zero-sum-games. Ngoài ra, giá trị mất mát của bộ sinh theo chuẩn 1 (đường nét chấm đứt xanh ngọc) cũng được giảm, tuy không đáng kể so với những mất mát còn lại.

Tuy hợp lí là thế, nhưng mô hình kiểu GAN khá khó để nói trước ngoài việc xem xét kết quả thử nghiệm. Sau một số thử nghiệm định tính đơn giản, mô hình có kết quả tốt nhất theo quan điểm chủ quan là mô hình sau epoch thứ 15³. Và sẽ là lựa chọn cho các thử nghiệm và đánh giá trong những phần sau.

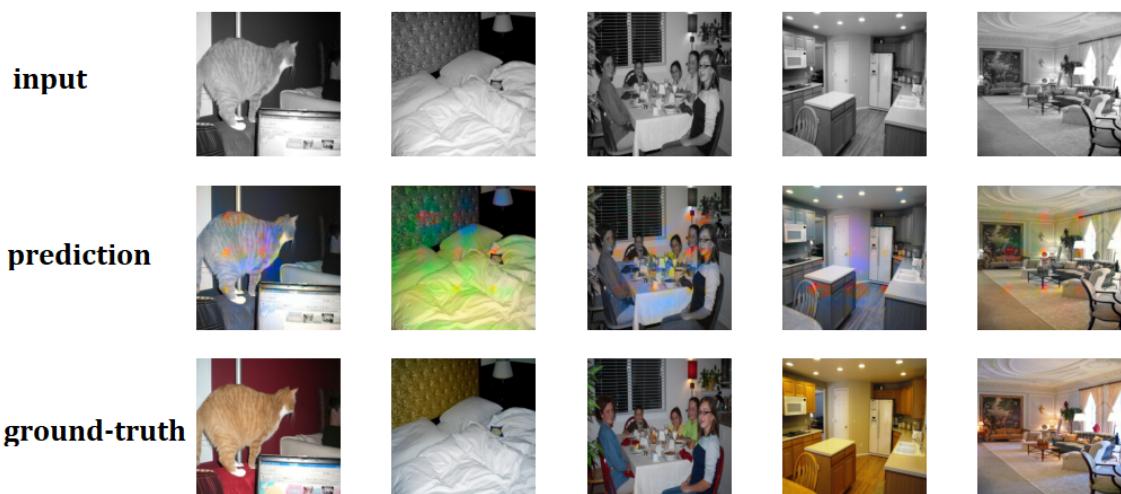
³Tất cả 20 mô hình qua mỗi epoch đều được lưu giữ lại trong thư mục đồ án tại đường dẫn [./training_results/GAN_models](#)

Chương 7

Thử nghiệm và đánh giá mô hình

7.1 Mô hình GAN với bộ sinh kiến trúc pix2pix

Kết quả ở hình dưới là kết quả của mô hình với bộ sinh xây dựng theo kiến trúc đề xuất của tác giả pix2pix [5.2], được thử nghiệm với 5 bức ảnh ngẫu nhiên lấy từ tập COCO (không nằm trong tập huấn luyện). Khá không may mắn, kết quả không được như chúng ta mong đợi (hình 7.1.1).

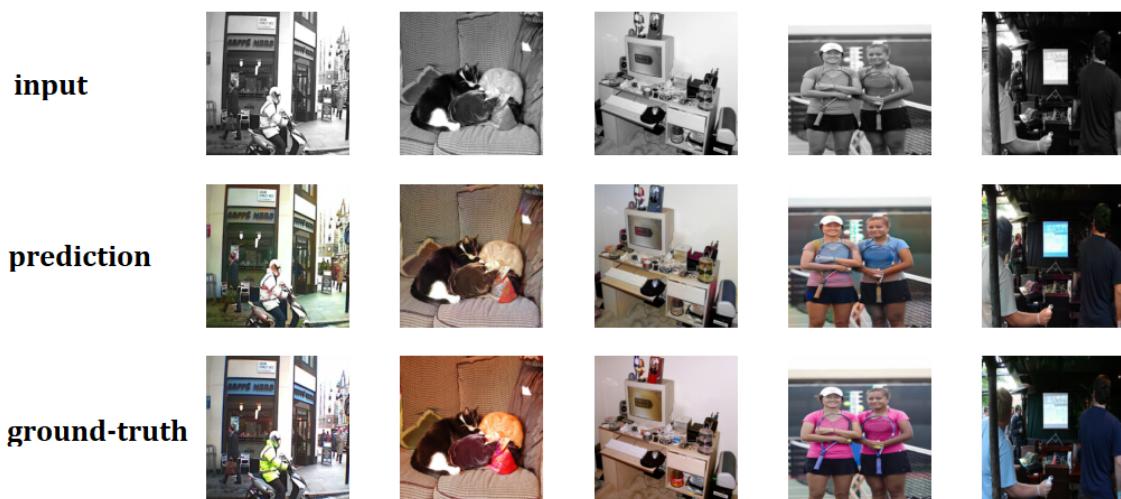


Hình 7.1.1: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản sau epoch thứ 34 trên tập dữ liệu COCO.

Nhìn chung, việc huấn luyện mô hình là khả thi khi mô hình cũng đã có thể tạo được màu cho một số điểm ảnh (hình 7.1.1 ở vị trí phía phải ngoài cùng) sau vài chục epoch. Nhưng như đã nêu ra [5.2], giới hạn về phần cứng cũng như là tập dữ liệu không đủ nhiều và tốt, để có thể huấn luyện được mô hình theo như mong đợi.

Nhìn thêm ở một góc độ khác, việc huấn luyện một mạng GAN cũng không hề dễ dàng, khi GAN rất dễ bị nhạy cảm với những thông số được cài đặt khi huấn luyện [20]. Sau nhiều lần thử nghiệm thất bại, một trong số những kinh nghiệm rút ra là, trong hầu hết các mô hình GAN, ta nên khởi tạo các trọng số của mô hình và bias theo phân phối $\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 0.1)$ để quá trình huấn luyện nhanh hội tụ hơn. Dĩ nhiên không thể thiếu một điều kiện cần là một tập dữ liệu có chất lượng tốt.

7.2 Mô hình GAN với bộ sinh có xương sống ResNet18



Hình 7.2.1: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản có xương sống ResNet18 sau epoch thứ 15 trên tập dữ liệu COCO (không nằm trong tập huấn luyện).

Kết quả ở hình 7.2.1 dễ thấy đã cải thiện hơn hẳn so với kết quả ở hình 7.1.1. Màu tạo ra ít bị lem hay lộn xộn như kết quả [7.1]. Một vài đối tượng có màu khác với màu của ảnh gốc, ví dụ người lái xe ở ảnh trái ngoài cùng, nhưng vẫn hết sức hợp lí và tự nhiên.

Ở một số vùng ảnh, mô hình có vẻ như vẫn chưa thực sự chọn được màu phù hợp và rồi để màu gần giống với mức xám ban đầu. Có thể là do những vùng chưa có màu hợp lí là trong quá trình huấn luyện, mô hình chưa được học qua ảnh có đặc trưng, đối tượng tương tự.



Hình 7.2.2: Mô hình hoàn toàn bế tắc trước logo kỷ niệm 60 năm của trường ĐH Bách Khoa TP.HCM.

Điều trên dẫn đến một hạn chế khác nữa là đối với những ảnh không có đặc trưng rõ ràng, những đặc trưng lạ lẫm mà chưa được đưa vào huấn luyện cho mô hình. Chẳng hạn ảnh chủ đề logo (hình 7.2.2), mô hình hoàn toàn không có khả năng tạo được màu phù hợp vì không hiểu được đặc trưng.

Thêm một hạn chế khác là mô hình cũng chưa thực sự biết được những điểm ảnh nào sẽ cùng một đối tượng và cùng màu. Cụ thể như trong hình 7.2.1, ảnh hai cô gái cầm vợt tennis



input



prediction

Hình 7.2.3: Trường hợp đặc biệt khi mô hình làm việc tệ với ảnh có đặc trưng đơn giản.

(thứ 2 từ phía bên phải), phần dưới của áo được chọn là màu hồng, còn phần trên của áo lại màu xanh. Nếu mô hình lựa cho phần áo màu hồng hay màu xanh thì đều chấp nhận được, nhưng mô hình lại chọn cách lựa hai phía của áo bởi hai màu khác nhau không đồng nhất.

Riêng với trường hợp hình 7.2.3 thực sự là một kết quả bất ngờ, mà hiện tại vẫn chưa tìm ra lí do thích hợp cho việc mô hình cho kết quả không thực sự tốt. Việc mô hình không có khả năng làm việc với đặc trưng mặt người là không khả thi, khi [7.3, 7.4] cũng có các ảnh mặt người tương tự và mô hình vẫn cho kết quả ổn. Đây là một trường hợp đặc biệt cần thêm nhiều tìm hiểu.

7.3 So sánh bộ sinh có xương sống ResNet18 trước và sau khi huấn luyện đối nghịch

Việc so sánh mô hình bộ sinh trước (bộ sinh đã được qua tiền huấn luyện đối lập) và sau khi huấn luyện đối nghịch cùng bộ phân biệt, sẽ cho ta thấy hiệu quả của huấn luyện kiểu GAN mang lại. Một số ảnh đen trắng (hình 7.3.1) được lựa chọn gồm các bối cảnh chân dung, phong cảnh, toà nhà cũng như những ảnh có nhiều đối tượng (chi tiết) để thử nghiệm.



Hình 7.3.1: Những tấm ảnh đen trắng được lựa chọn để so sánh sự khác biệt.

Trước khi đưa cho mô hình thực hiện tạo màu, không khó để đoán được rằng hình ảnh đầu tiên (phía trái ngoài cùng) có ít chi tiết nhất nên khả năng cao sẽ có kết quả tốt nhất. Ngược lại, hình ảnh cuối cùng (phía phải ngoài cùng) có rất nhiều chi tiết nên kết quả cũng rất có thể

không đạt được tốt như những tấm ảnh ít chi tiết. Và quả thật không ngoài dự đoán, kết quả có được khá đúng với những gì mong đợi.



Hình 7.3.2: Phía trên và dưới lần lượt là kết quả của bộ sinh trước và sau khi huấn luyện đối nghịch.

Ta có thể thấy trong hình 7.3.2 một điều là, mô hình sau khi được huấn luyện theo kiểu GAN cho kết quả có màu tươi sáng, chân thật hơn so với khi chưa được uốn nắn bởi bộ phân biệt. Kết quả đã cải thiện một cách rõ rệt, nên khó có thể nói là mô hình đối nghịch trong mạng GAN không có đóng góp mà chỉ do bộ sinh đã được huấn luyện thêm.

Điểm yếu của việc áp dụng đơn giản hóa mất mát so sánh sai lệch giữa các điểm ảnh, bằng cách dùng chuẩn bậc 1 hoặc bậc 2 như thông thường, đối với bài toán tạo màu rất khó để có kết quả tốt. Vì mô hình lúc này được khuyến khích để dùng màu xám [87]. Rõ nghĩa hơn, mô hình đã học được một quy tắc, khi không chắc chắn nên chọn màu gì, thì chọn một giá trị trung bình trong thang $[-1, 1]$ là $\approx (0, 0)$ là một cách an toàn nhất để có thể tối thiểu sai số. Dẫn đến màu tạo ra có xu hướng không khác là mấy so với màu xám. Và dĩ nhiên màu trong cuộc sống của chúng ta rất nhiều màu sắc sặc sỡ hơn màu xám. Khiến cho kết quả của mô hình trước khi đưa vào mạng GAN huấn luyện chưa đáp ứng mục tiêu hợp lý, chân thực đưa ra.

7.4 Khảo sát đánh giá định tính chất lượng mô hình

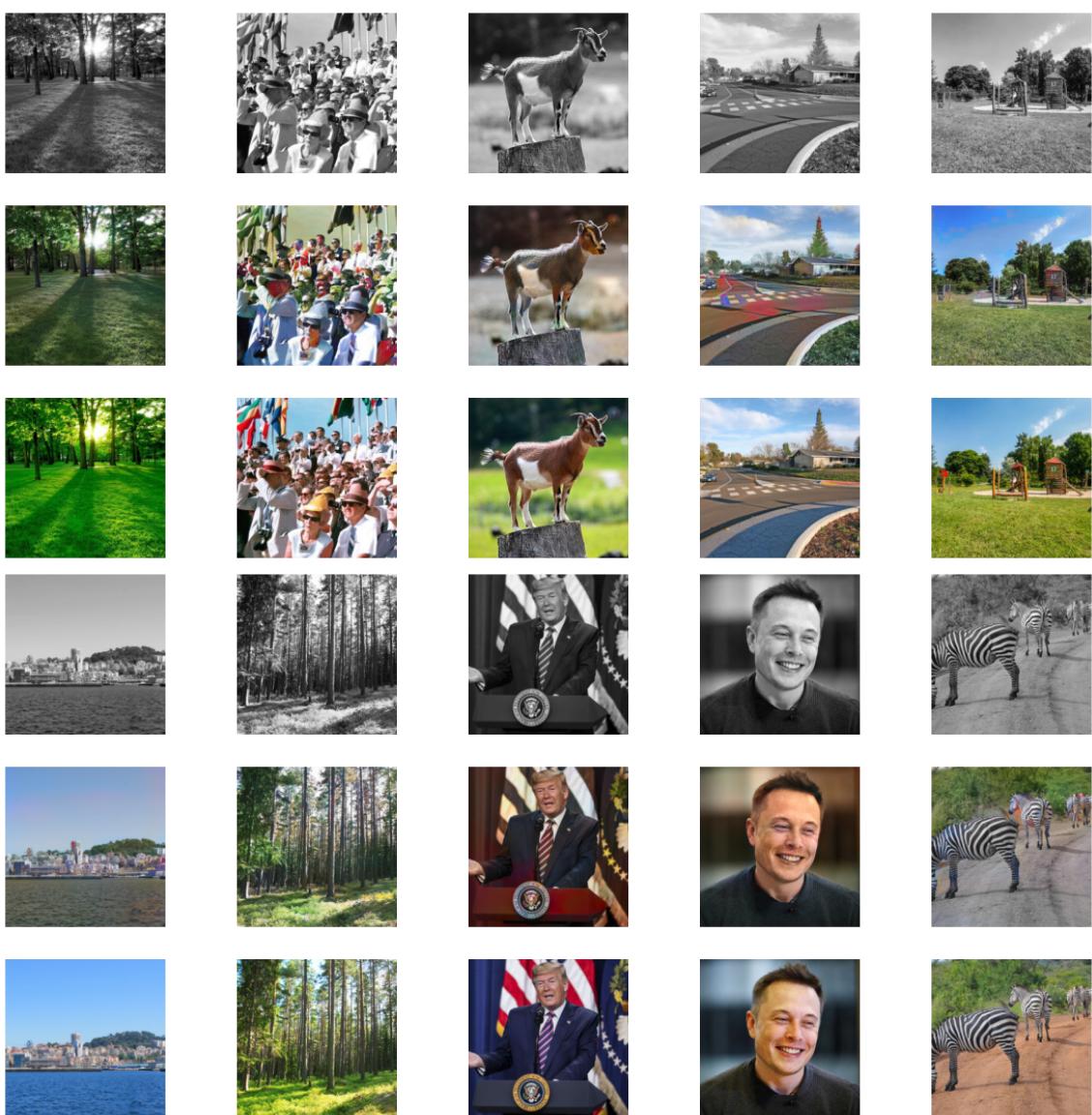
Với nhiều bài toán liên quan về đồ họa, hình ảnh điển hình như bài toán tạo màu, một bài toán có nhiều lời giải thì một trong những bài kiểm tra tốt nhất là kiểm tra dưới góc độ của thị giác con người. Một cuộc khảo sát (hình 7.4.3) đã được tiến hành bởi sự giúp đỡ của 100 người, bao gồm những sinh viên cũng như không thuộc Đại học Quốc gia và một số học sinh cấp trung học phổ thông. Bài kiểm tra đánh giá bao gồm 20 bức ảnh như trong hình 7.4.1 và hình 7.4.2, được đánh số thứ tự từ trái sang phải, trên xuống dưới. Thang điểm đưa ra là từ 1 (rất giả) tới 5 (rất thật). Trong phần đánh giá, chỉ có ảnh sau khi tạo màu - ảnh ở hàng thứ hai, để tránh thiên lệch khi đánh giá từ người tham gia (hình 7.4.4).

Bảng 7.4.1 là kết quả bình chọn từ khảo sát 100 người. Không quá khó khăn để tính được điểm trung bình mà một tấm ảnh nhận được, bằng cách lấy tổng của số lượt bình chọn nhân với thang điểm tương ứng, rồi chia cho tổng số bình chọn (ở đây là 100). Kết quả tính toán trên được cho ở bảng 7.4.2.

Trong những ảnh được bình chọn, ảnh có số điểm cao nhất là ảnh **05** với số điểm trung bình là **4.01** (mức 4/5). Theo sau đó là những ảnh có số điểm 3.96 (07) và 3.94 (09) (hình

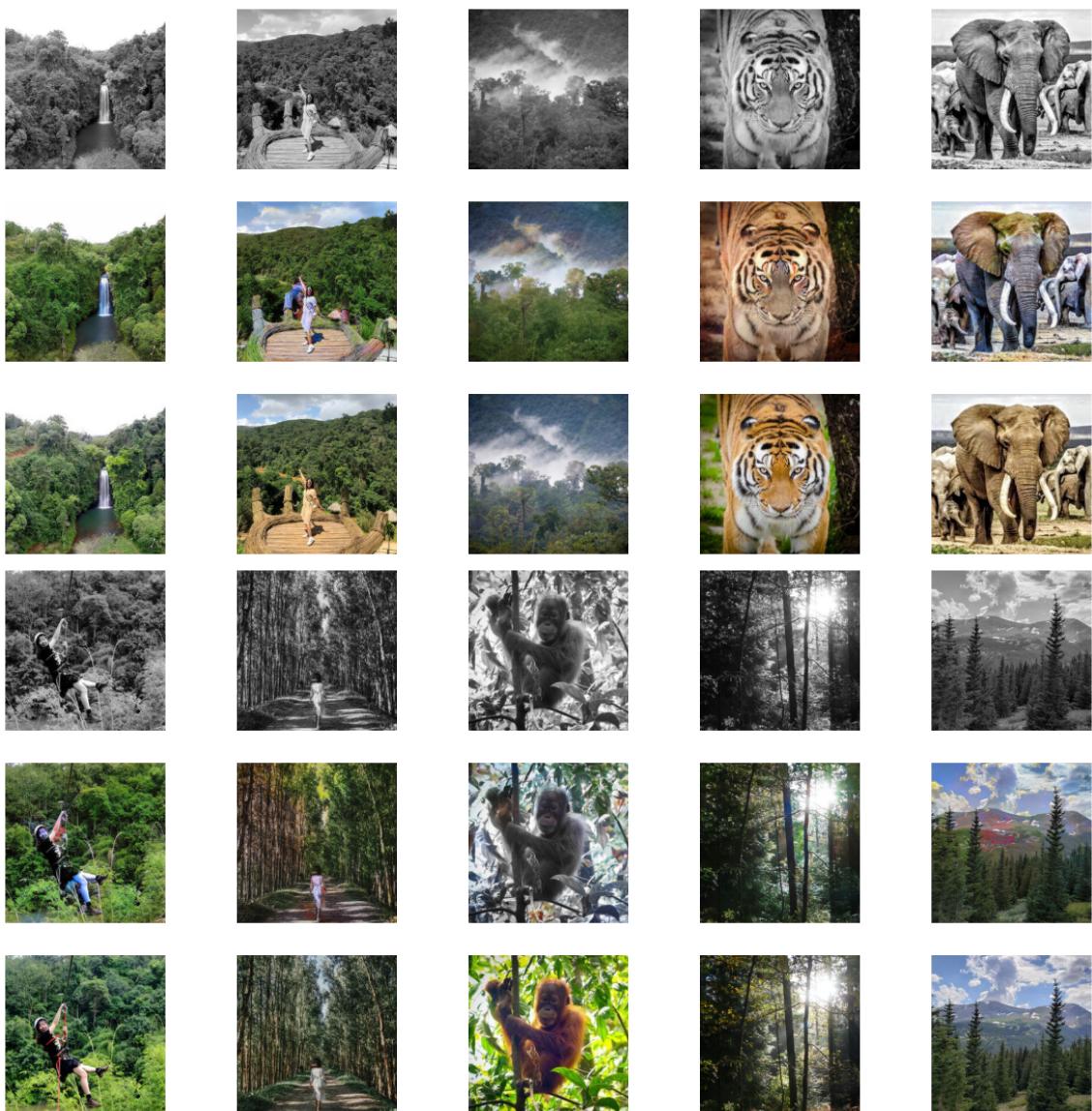
7.4.5). Còn ảnh có số điểm trung bình thấp nhất là ảnh 02 với **2.53** (mức 2–3/5). Ngay phía trên là những ảnh có số điểm 2.75 (15) và 2.76 (20) (hình 7.4.6). Điểm trung bình của 20 bức ảnh khảo sát là **3.38** (mức 3) và có **11 (55%)** ảnh có số điểm trung bình ở trên mức này.

Một điều dễ nhận ra là những ảnh có cây cối, phong cảnh thiên nhiên sẽ là những ảnh có điểm khá cao, từ mức 3 tới mức 5. Và 2 trong số 3 bức ảnh đạt số điểm cao nhất cũng đều có đặc trưng như trên. Duy chỉ với ảnh 20, khi có phần tó lem nên đã bị đánh giá thấp đi nhiều, dẫu những phần còn lại có kết quả màu được tạo ra rất đẹp. Với những ảnh động vật thì kết quả không được khả quan như vậy. Một trong số đó có kết quả gần tệ nhất là ảnh 15. Chỉ riêng mỗi ảnh 03 có số điểm khá vượt trội so với những ảnh cùng đặc trưng. Với đặc trưng về con người, nếu không có quá nhiều chi tiết thì mô hình thể hiện khá tốt và cho số điểm rất cao (3.86 cho ảnh 08 và 3.94 cho ảnh 09). Nhưng với nhiều chi tiết thì mô hình lại không giữ được phong độ và cho kết quả cực kì tệ, như ảnh 16 với số điểm 2.91 và ảnh 02 với số điểm thấp nhất trong 20 bức ảnh khảo sát.



Hình 7.4.1: Một số kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc.

¹Google Forms, <https://www.google.com/forms/about/>



Hình 7.4.2: Thêm Một vài kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc.

Đánh giá mức độ hợp lý, chân thật của màu ảnh nhân tạo

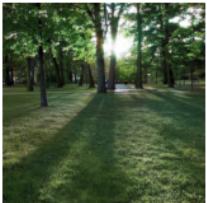
Mình đã xây dựng một chương trình để tạo màu cho ảnh xám. Dưới đây là một số ảnh đã được chương trình của mình tạo màu. Mong bạn đánh giá độ hợp lý, chân thật của màu ảnh theo thang điểm từ 1 (rất giả) tới 5 (rất thật).

Rất cảm ơn bạn đã bỏ công sức giúp mình.

***Bắt buộc**

Hình 7.4.3: Phần mô tả của bài đánh giá.

Ảnh 01 *



1 2 3 4 5

rất giả rất thật

Questions Responses **100**

100 responses

Not accepting responses

Message for respondents

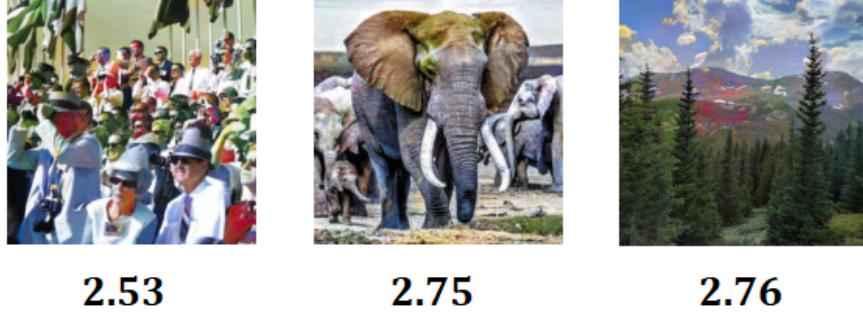
This form is no longer accepting responses

Summary Question Individual

Hình 7.4.4: Phần câu hỏi và số lượng người tham gia cuộc khảo sát thực hiện thông qua Google Forms¹.



Hình 7.4.5: Những ảnh có điểm trung bình cao nhất.



Hình 7.4.6: Những ảnh có điểm trung bình thấp nhất.

Thang điểm	1	2	3	4	5	Tổng
Ảnh						
Ảnh 01	2	11	29	36	22	100
Ảnh 02	20	35	24	14	7	100
Ảnh 03	1	11	21	34	33	100
Ảnh 04	8	34	30	18	10	100
Ảnh 05	2	7	18	34	39	100
Ảnh 06	9	24	37	21	9	100
Ảnh 07	1	12	15	34	38	100
Ảnh 08	1	11	21	35	32	100
Ảnh 09	1	10	21	30	38	100
Ảnh 10	8	16	36	29	11	100
Ảnh 11	4	8	19	33	36	100
Ảnh 12	2	18	28	26	26	100
Ảnh 13	7	18	22	34	19	100
Ảnh 14	6	14	24	34	22	100
Ảnh 15	13	27	40	12	8	100
Ảnh 16	12	32	23	19	14	100
Ảnh 17	14	26	30	18	12	100
Ảnh 18	10	22	26	27	15	100
Ảnh 19	3	8	23	29	37	100
Ảnh 20	22	22	26	18	12	100
Tổng	146	366	513	535	440	2000

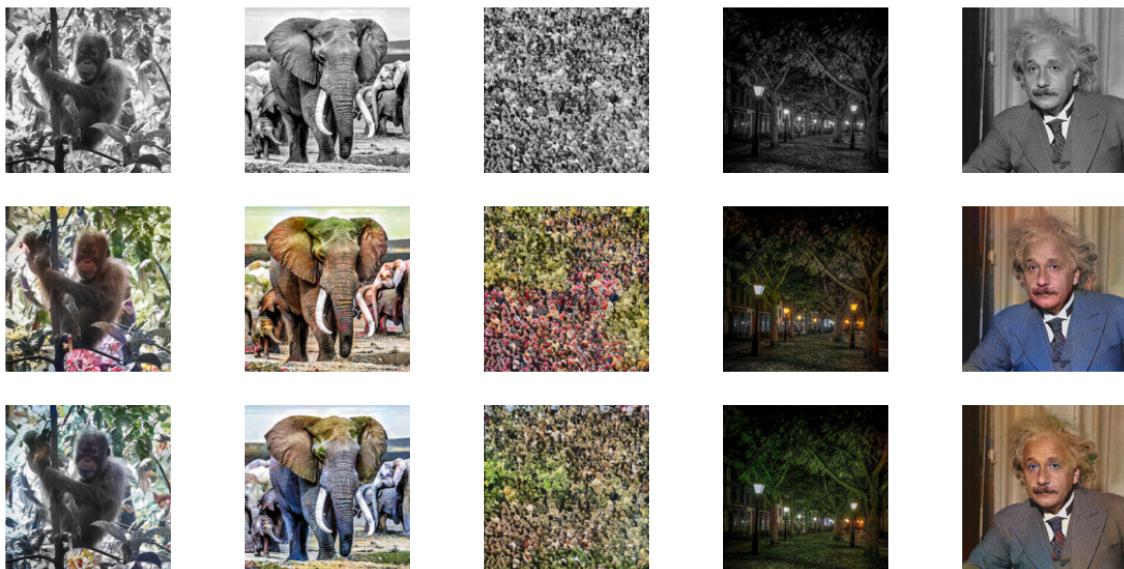
Bảng 7.4.1: Kết quả bình chọn cho các ảnh.

Điểm trung bình	Ảnh 01	Ảnh 02	Ảnh 03	Ảnh 04	Ảnh 05
	3.65	2.53	3.87	2.88	4.01
Điểm trung bình	Ảnh 06	Ảnh 07	Ảnh 08	Ảnh 09	Ảnh 10
	2.97	3.96	3.86	3.94	3.19
Điểm trung bình	Ảnh 11	Ảnh 12	Ảnh 13	Ảnh 14	Ảnh 15
	3.89	3.56	3.40	3.52	2.75
Điểm trung bình	Ảnh 16	Ảnh 17	Ảnh 18	Ảnh 19	Ảnh 20
	2.91	2.88	3.15	3.89	2.76

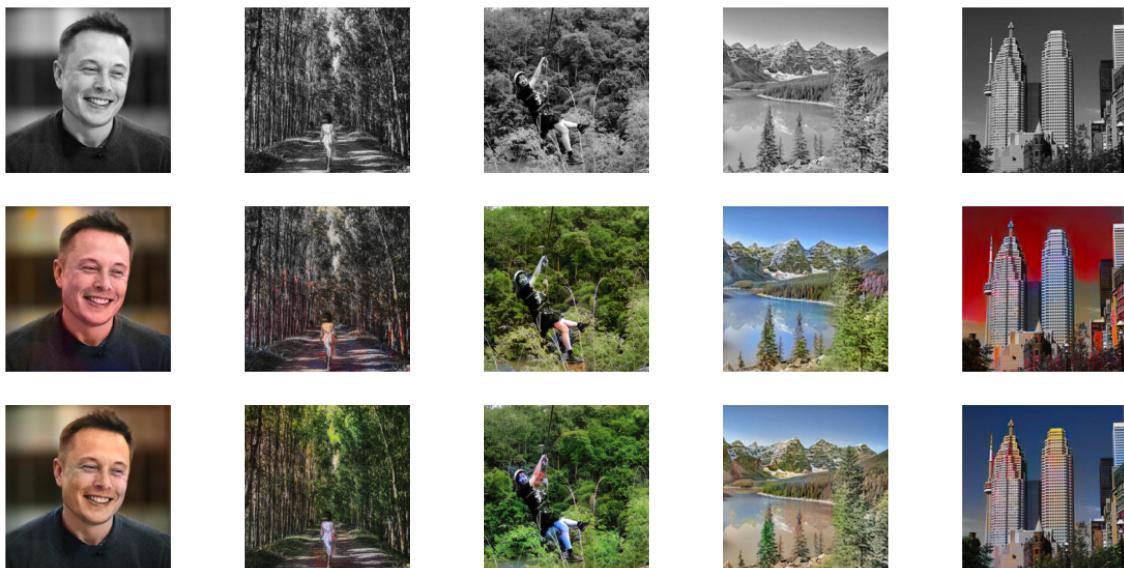
Bảng 7.4.2: Điểm trung bình của các ảnh.

Tuy số lượng mẫu khảo sát nhỏ với chỉ 20 bức ảnh (không đủ làm tổng thể) và sự đa dạng chưa cao, cũng như là số lượng người tham gia khảo sát không quá lớn - chỉ với 100 người. Nhưng cũng phần nào đánh giá được chất lượng của mô hình. Nhìn chung, với số điểm trung bình 3.38 và số lượng người đồng tính mức 4 (mức khá) cũng đồng đảo nhất với **535 (26.75%)** (bảng 7.4.1) người, thì mô hình có thể xem ở mức **trung bình-khá**.

7.5 So sánh kết quả sau khi huấn luyện lại với tập dữ liệu lớn hơn



Hình 7.5.1: So sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại.



Hình 7.5.2: Thêm một vài so sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại.

Dây là mô hình được khởi tạo theo kiến trúc và các thông số hoàn toàn giống với mô hình mà ta đã thử nghiệm [7.4]. Mô hình được huấn luyện với một tập dữ liệu giàu hơn, gồm 18,000 bức ảnh (so với 10,000 của mô hình cũ) từ tập dữ liệu COCO. Số lượng đã tăng gấp 1.8 lần, kết hợp phương pháp làm giàu dữ liệu thì số lượng gấp 3.6 lần.

Việc huấn luyện với số lượng dữ liệu nhiều hơn, giúp cho mô hình có thêm nhiều hiểu biết về các đặc trưng. Từ đó, một số kết quả trở nên chân thực hơn (hình 7.5.1, cột 2; hình 7.5.2, cột 3).

Một sự khác biệt ở mô hình này, là quá trình huấn luyện không qua bước tiền huấn luyện bộ sinh. Có lẽ cũng vì vậy, mà màu của mô hình này tươi sáng hơn rất nhiều, khi có nhiều điểm ảnh được cho màu đỏ hoặc lam (màu áo ở hình 7.5.1, cột thứ 5). Việc dũng cảm hơn trong việc tô màu, thay vì chọn màu xám, giúp cho một vài thứ có kết quả đẹp hơn (màu đèn ở hình 7.5.1, cột thứ 4; màu nước ở hình 7.5.2, cột thứ 4). Nhưng cũng đầy rủi ro, khi lại dễ dàng gây ra sự lem luốc (màu trời, màu toà nhà ở hình 7.5.2, cột thứ 5). Cũng giống với vua Midas, ta cần phải hiểu rõ cái ta muốn. Để cân đối được chuyện sáng tạo cho mô hình, công việc này cần thêm nhiều thử nghiệm trong tương lai.

7.6 Đánh giá định lượng hai mô hình trước và sau khi cải thiện bằng MSE và RMSE

Mô hình trước cải thiện \mathcal{G}_1 là mô hình được huấn luyện với 10,000 ảnh. Sau cải thiện là mô hình \mathcal{G}_2 được huấn luyện với 18,000 ảnh. Ta sẽ đánh giá hai mô hình qua MSE [80] và RMSE [83]. Công thức để đánh giá MSE sử dụng được cho ở (7.1), còn RMSE là căn bậc 2 của MSE (7.2).

Ta sẽ khảo sát 2 ảnh kênh màu, tức là $\widehat{\mathcal{I}}_{\text{đầu ra } \mathcal{G}_1} = \mathcal{G}_1(\mathcal{I}_{\text{đầu vào}})$ với $\widehat{\mathcal{I}}_{\text{đầu ra } \mathcal{G}_2} = \mathcal{G}_2(\mathcal{I}_{\text{đầu vào}})$ so với $\mathcal{I}_{\text{đầu ra}}$. Ở đây, $\mathcal{I}_{\text{đầu ra}}, \mathcal{I}_{\text{đầu ra } \mathcal{G}_1}, \mathcal{I}_{\text{đầu ra } \mathcal{G}_2}$ sẽ được chuẩn hoá theo tập $[-110, 110] \times [-110, 110]$.

$$\text{MSE} = \frac{2}{256 \times 256} \sum_{k=1}^2 \sum_{i=1}^{256} \sum_{j=1}^{256} \left\{ \widehat{\mathcal{I}}_{\text{đầu ra}} - \mathcal{I}_{\text{đầu ra}} \right\}^2 \quad (7.1)$$

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{2}{256 \times 256} \sum_{k=1}^2 \sum_{i=1}^{256} \sum_{j=1}^{256} \left\{ \widehat{\mathcal{I}}_{\text{đầu ra}} - \mathcal{I}_{\text{đầu ra}} \right\}^2} \quad (7.2)$$

	Ảnh 02 (2.53)	Ảnh 07 (3.96)	Ảnh 09 (3.94)	Ảnh 15 (2.75)
MSE \mathcal{G}_1	374.2613	312.994	240.9714	628.1626
MSE \mathcal{G}_2	409.466	202.8153	150.6662	442.2453
RMSE \mathcal{G}_1	19.3458	17.6916	15.5233	25.0632
RMSE \mathcal{G}_2	20.2353	14.2413	12.2746	21.0296

Bảng 7.6.1: Kết quả MSE và RMSE của hai mô hình trên 4 ảnh được chọn.

Có 4 ảnh được đem ra để đánh giá. Trong đó, một nửa là 2 tấm trong những ảnh có kết quả tệ (ảnh 02, ảnh 15), còn lại là 2 tấm trong nhóm kết quả tốt (ảnh 07, ảnh 09) [7.4]. Kết quả được cho ở bảng 7.6.1.

Một điều dễ thấy đó là, mô hình \mathcal{G}_2 cho kết quả tốt hơn so với \mathcal{G}_1 khi 3/4 ảnh (ảnh 07, ảnh 09 và ảnh 15) có kết quả MSE thấp hơn. Ảnh 02 là một ảnh nhiều chi tiết, với tính cách tinh nghịch của \mathcal{G}_2 khi không được tiền huấn luyện, rất dễ để có những màu lêch so với màu gốc. Nên MSE của \mathcal{G}_2 cao hơn so với của \mathcal{G}_1 .

Thông qua RMSE, ta có thể thấy được rằng, mỗi điểm màu của cả hai mô hình sẽ chênh lệch khoảng 10 đến 20 (thấp nhất là 12.2746 của \mathcal{G}_2 với ảnh 09, nhiều nhất là 25.0632 của \mathcal{G}_1 với ảnh 15) đơn vị so với ảnh gốc. Kết quả này quả thực không quá tệ khi chỉ chênh khoảng $100\% \times \frac{10}{2 \times 110} = 4.55\%$ đến $100\% \times \frac{20}{2 \times 110} = 9.09\%$ so với khoảng giá trị.

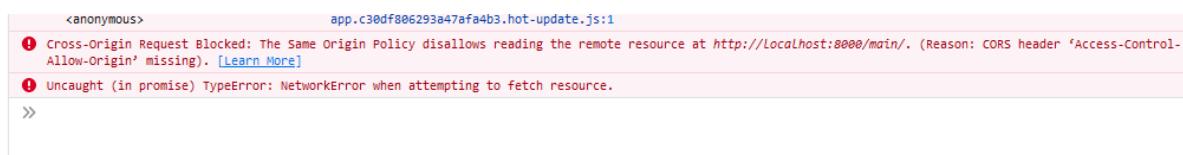
Chương 8

Ứng dụng

8.1 Giao diện lập trình ứng dụng (API - Application Programming Interface) bằng khung phần mềm Django của Python

Để có thể dễ dàng chia sẻ tính năng phần mềm, cụ thể là việc tạo màu, API là một phương thức hữu hiệu được sử dụng nhiều nhất hiện nay. Ngoài ra, với việc mô hình được triển khai bằng ngôn ngữ Python, một ngôn ngữ có một khung phần mềm lập trình liên quan về website mạnh mẽ như Django, là một lợi thế.

Dựa trên những cơ sở đó, một API tạo màu đã được xây dựng có endpoint là `domain/main/` (trong quá trình thử nghiệm, domain là `localhost:8000`). Phương thức gửi yêu cầu được chấp nhận là **POST**. Nội dung ảnh sẽ được đính kèm, không qua mã hoá, trong phần thân của yêu cầu gửi tới (hình 8.1.3). Vì mục đích thử nghiệm là chủ yếu, các phương thức bảo mật CSRF, CORS (hình 8.1.1) được tắt để tránh gây phiền hà khi thao tác.

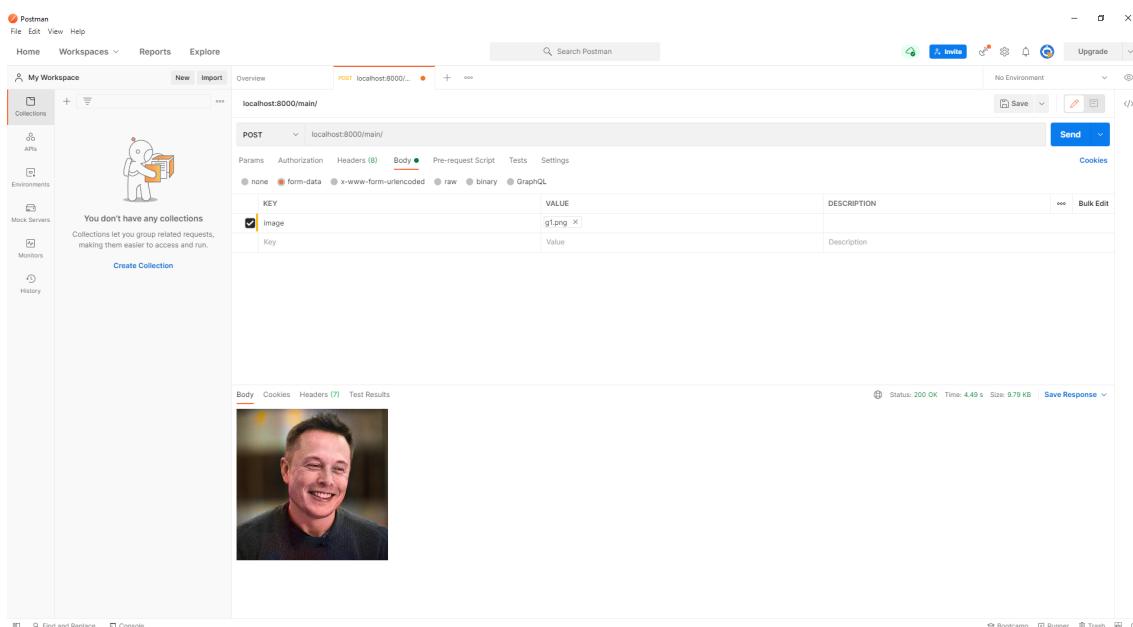


Hình 8.1.1: Bảo mật CORS cản trở khi thử nghiệm API

```
<anonymous>          app.c30df8806293a47afa4b3.hot-update.js:1
  Response { type: "cors", url: "http://localhost:8000/main/", redirected: false, status: 200, ok: true, statusText: "OK", headers: Headers, body: ReadableStream, bodyUsed: false }
    body: ReadableStream { locked: false }
      locked: false
        <prototype>: ReadableStream.prototype { cancel: cancel(), getReader: getReader(), tee: tee(), ... }
        cancel: function cancel()
        constructor: function ReadableStream()
        getReader: function getReader()
        locked: >
        tee: function tee()
        Symbol(Symbol.toStringTag): "ReadableStream"
        <get locked()>: function locked()
        <prototype>: Object { ... }
      bodyUsed: false
    headers: Headers { }
      <prototype>: HeadersPrototype { append: append(), delete: delete(), get: get(), ... }
      ok: true
      redirected: false
      status: 200
      statusText: "OK"
      type: "cors"
      url: "http://localhost:8000/main/"
      <prototype>: ResponsePrototype { clone: clone(), arrayBuffer: arrayBuffer(), blob: blob(), ... }
```

Hình 8.1.2: Phản hồi trả về của endpoint

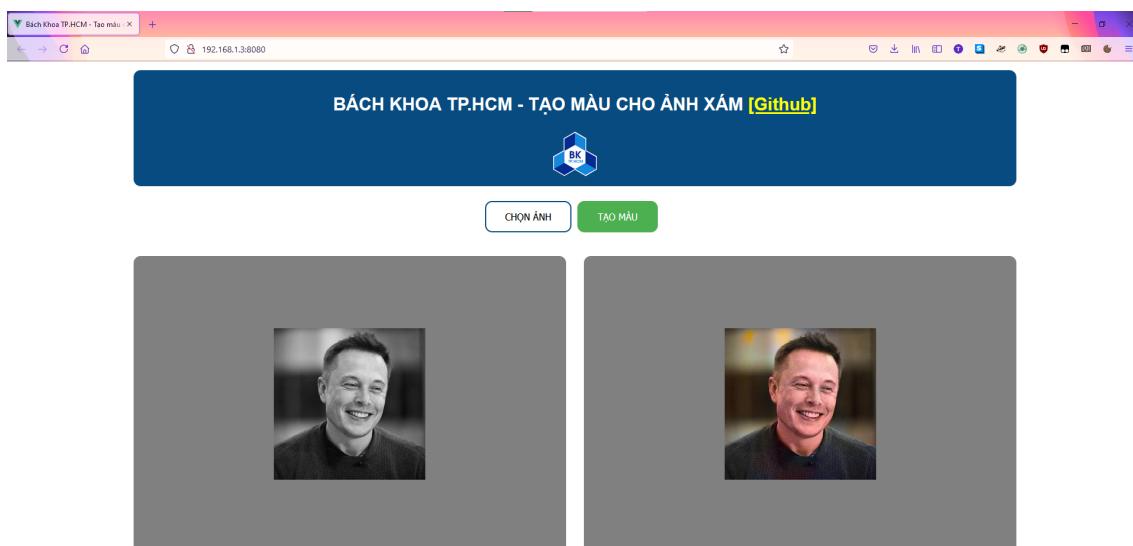
Khi nhận được yêu cầu, endpoint tiếp nhận và xử lý, sau đó trả về phản hồi đính kèm ảnh được tạo màu dưới định dạng dữ liệu ReadableStream (hình 8.1.2).



Hình 8.1.3: Thử nghiệm API trên phần mềm Postman

8.2 Ứng dụng web với khung phần mềm VueJS của Javascript và API

Tận dụng API đã được xây dựng, kết hợp với khung phần mềm thiết kế giao diện VueJS, ta dễ dàng tạo được một ứng dụng website thân thiện với người dùng (hình 8.2.1). Chỉ cần chọn một tấm ảnh xám, ứng dụng trả về một tấm ảnh sau khi được tạo màu và cho phép người dùng tải về.



© Nguyễn Thành Trung - 1814515 (trung.nguyendx@hcmut.edu.vn)

Hình 8.2.1: Ứng dụng web tạo màu đơn giản bằng VueJS kết hợp API

Chương 9

Tổng kết

9.1 Kết luận

Một phương pháp tạo màu cho ảnh xám đã được trình bày thông qua đồ án này. Phương pháp sử dụng khung mô hình một mạng đối nghịch tạo sinh có điều kiện - cGAN theo khung mô hình pix2pix. Mạng gồm bộ phân biệt kiến trúc Patch 70×70 , và một bộ sinh mạng tích chập kiến trúc Unet có phần xương sống học chuyển giao từ mạng tích chập ResNet18.

Mô hình được lập trình và huấn luyện bằng ngôn ngữ Python, trên nền tảng mã nguồn mở về máy học Pytorch của Facebook's AI Research lab. Quá trình huấn luyện cho ra hai mô hình. Mô hình đầu tiên được huấn luyện theo hàm mất mát kiểu GAN kết hợp với một hàm mất mát truyền thống chuẩn 1, trên tập dữ liệu 10,000 bức ảnh của COCO (kết hợp với làm giàu dữ liệu là 20,000 bức ảnh). Mô hình thứ hai là mô hình cải thiện với các thông số khởi tạo giống với mô hình ban đầu, nhưng hàm mất mát không còn kết hợp hàm mất mát chuẩn 1. Tập dữ liệu cũng được mở rộng hơn với 18,000 bức ảnh của COCO (kết hợp với làm giàu dữ liệu là 36,000 bức ảnh).

Chất lượng mô hình đầu tiên được đánh giá dựa qua một cuộc khảo sát thực tế, với sự tham gia giúp đỡ của 100 người. Bài khảo sát bao gồm 20 bức ảnh có màu tạo từ bộ sinh, theo thang điểm từ 1 (rất giả) tới 5 (rất thật). Kết quả nhận được là số điểm trung bình 3.38, trong đó số điểm cao nhất là 4.01 và thấp nhất là 2.53. Sau khi cải thiện mô hình, kết quả MSE và RMSE đã cho thấy mô hình sau đã được cải thiện hơn so với mô hình trước. Theo như mục tiêu được đề ra, cả hai mô hình đã hoàn thành được ở mức trung bình-khá cho tới khá.

Bên cạnh đó, cũng đã thành công xây dựng một API tạo màu bằng khung phần mềm Django của Python. Từ đó triển khai một ứng dụng web đơn giản để tạo màu cho ảnh bằng khung phần mềm VueJS của Javascript.

Đề tài của đồ án đã được hoàn thành.

9.2 Hạn chế và hướng cải thiện mô hình

Mô hình vẫn chưa có kết quả tốt khi áp dụng với những ảnh có một số đặc trưng lạ, những đặc trưng chưa được học trong quá trình huấn luyện. Để cải thiện trí tuệ mô hình, ta cần tìm thêm nhiều tập dữ liệu với các đối tượng, đặc trưng phong phú để huấn luyện thêm. Ngoài ra, sử dụng một xương sống tốt hơn mạng ResNet18 hiện có, chẳng hạn ResNet34, ResNet50, ResNet101, hoặc một họ mạng khác như VGG hay Xception.

Một cách tiếp cận khác để cải thiện mô hình, đó là huấn luyện nhiều bộ sinh. Mỗi bộ sinh sẽ đảm nhiệm một nhóm đặc trưng nhất định. Như vậy chỉ cần cho bộ sinh học những ảnh liên

quan đến chủ đề của mình. Bằng việc chuyên môn hoá chủ đề cho bộ sinh, ta không còn yêu cầu một bộ sinh có thể tạo màu tốt với mọi ảnh đầu vào. Thay vào đó, cho kết quả tốt hơn cả khi được đưa cho ảnh đúng chuyên môn. Quá trình huấn luyện cũng được đơn giản hơn do mô hình không phải học quá nhiều.

Một số những đề xuất về việc sử dụng GAN để tạo màu có kết quả tốt có thêm tham khảo thêm như [41], khi tác giả sử dụng mạng GAN tích chập sâu - DCGAN áp dụng với ảnh độ phân giải cao, tăng tốc độ và ổn định huấn luyện. Cao và các cộng sự của mình [9] cũng sử dụng cGAN để tạo màu một cách đa dạng nhờ vào lấy mẫu nhiều đầu vào nhiều lần. Một đột phá khác, Vitoria [63] nhận thấy rằng ta có tận dụng phân phối phân lớp trong quá trình huấn luyện cho mạng GAN, và đưa ra mô hình với cái tên là ChromaGAN.

9.3 Hướng phát triển đề tài

Đề tài có thể được áp dụng để làm những tác vụ liên quan đến phục chế màu cho ảnh xám, bị mất màu hoặc ảnh cũ xưa, thời chiến tranh. Giúp cho quá trình phục chế được tiết kiệm thời gian. Phối hợp với một ít tinh chỉnh bằng phương pháp thủ công, ta hoàn toàn có thể có những kết quả phục chế đẹp đẽ và tiết kiệm được rất nhiều thời gian.

Không chỉ dừng lại ở hình ảnh đơn lẻ, ta cũng có thể áp dụng mô hình để phục chế màu cho các video, vì bản chất một video là sự kết hợp từ nhiều khung hình ghép lại. Trong thư mục đồ án, phần `test_results` có một kết quả thử nghiệm của mô hình đầu tiên trong việc phục chế màu một đoạn nhỏ về phóng sự về chiến tranh ở Việt Nam.



Hình 9.3.1: Ảnh Trường ĐH Bách Khoa TP.HCM ngày xưa khi chưa có màu (trên) và sau khi tô màu: mô hình ban đầu (trái), mô hình sau khi cải thiện (phải)

Tài liệu tham khảo

- [1] Nikolas Adaloglou. "Intuitive Explanation of Skip Connections in Deep Learning". In: <https://theaisummer.com/> (2020). URL: <https://theaisummer.com/skip-connections/>.
- [2] Ben Allison. "Reply: How and why do normalization and feature scaling work?" In: <https://stats.stackexchange.com/> (2012). URL: <https://bit.ly/34keB0w>.
- [3] Samuel A. Barnett. *Convergence Problems with Generative Adversarial Networks (GANs)*. 2018. arXiv: [1806.11382 \[cs.LG\]](https://arxiv.org/abs/1806.11382).
- [4] Ashwin Bhandare et al. "Applications of Convolutional Neural Networks". In: *International Journal of Computer Science and Information Technologies* 7 (2016), pp. 2206–2215.
- [5] The BrownBatman. "Reply: Calculate the Output size in Convolution layer". In: <https://stackoverflow.com/> (2018). URL: <https://bit.ly/3fn0Jrn>.
- [6] Jason Brownlee. "A Gentle Introduction to Imbalanced Classification". In: <https://machinelearningmastery.com/> (2019). URL: <https://bit.ly/3xasETz>.
- [7] Jason Brownlee. "How to Develop a Pix2Pix GAN for Image-to-Image Translation". In: <https://machinelearningmastery.com/> (2019). URL: <https://bit.ly/3hVPOrY>.
- [8] Jason Brownlee. "How to use Data Scaling Improve Deep Learning Model Stability and Performance". In: <https://machinelearningmastery.com/> (2019). URL: <https://bit.ly/2QQQ1QK>.
- [9] Yun Cao et al. *Unsupervised Diverse Colorization via Generative Adversarial Networks*. 2017. arXiv: [1702.06674 \[cs.CV\]](https://arxiv.org/abs/1702.06674).
- [10] Z. Cheng, Q. Yang, and B. Sheng. "Deep colorization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 415–423.
- [11] Community. "Reply: Why does expressing calculations as matrix multiplications make them faster?" In: <https://stats.stackexchange.com/> (2018). URL: <https://bit.ly/3hAcbkS>.
- [12] Grégoire Delétang. "How GANs really work". In: <https://towardsdatascience.com/> (2019). URL: <https://bit.ly/2U0AnIu>.
- [13] Bo Fu et al. *A Convolutional Neural Networks Denoising Approach for Salt and Pepper Noise*. 2018. arXiv: [1807.08176 \[cs.MM\]](https://arxiv.org/abs/1807.08176).
- [14] Richard Gall. "Working Principles of Generative Adversarial Networks (GANs)". In: <https://dzone.com/> (2018). URL: <https://bit.ly/3ue1M3r>.
- [15] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: [1406.2661 \[stat.ML\]](https://arxiv.org/abs/1406.2661).
- [16] Hyungrok Ham, Tae Joon Jun, and Daeyoung Kim. *Unbalanced GANs: Pre-training the Generator of Generative Adversarial Network using Variational Autoencoder*. 2020. arXiv: [2002.02112 \[cs.LG\]](https://arxiv.org/abs/2002.02112).

- [17] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: [1512.03385 \[cs.CV\]](#).
- [18] Gao Huang et al. *Densely Connected Convolutional Networks*. 2018. arXiv: [1608.06993 \[cs.CV\]](#).
- [19] Y.-C. Huang et al. “An adaptive edge detection based colorization algorithm and its applications”. In: *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM. 2005, pp. 351–354.
- [20] Jonathan Hui. “GAN — Why it is so hard to train Generative Adversarial Networks!” In: <https://jonathan-hui.medium.com/> (2018). URL: <https://bit.ly/3jKdnVB>.
- [21] R. Ironi, D. Cohen-Or, and D. Lischinski. “Colorization by example”. In: *Rendering Techniques*. Citeseer. 2005, pp. 201–210.
- [22] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: [1611.07004 \[cs.CV\]](#).
- [23] Jantic. *DeOldify*. 2019. URL: <https://github.com/jantic/DeOldify>.
- [24] Juan. “Reply: What range does skimage use in LAB color space for each channel?” In: <https://stackoverflow.com/> (2020). URL: <https://bit.ly/3hvscj>.
- [25] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. arXiv: [1812.04948 \[cs.NE\]](#).
- [26] Aditya Khamparia et al. “Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network”. In: *IEEE Access* 7 (2019), pp. 7717–7727. DOI: [10.1109/ACCESS.2018.2888882](#).
- [27] Asifullah Khan et al. “A survey of the recent architectures of deep convolutional neural networks”. In: *Artificial Intelligence Review* 53.8 (Apr. 2020), pp. 5455–5516. ISSN: 1573-7462. DOI: [10.1007/s10462-020-09825-6](#). URL: <http://dx.doi.org/10.1007/s10462-020-09825-6>.
- [28] Suleman Khan et al. “Facial Recognition using Convolutional Neural Networks and Implementation on Smart Glasses”. In: *2019 International Conference on Information Science and Communication Technology (ICISCT)* (2019). DOI: [10.1109/CISCT.2019.8777442](#).
- [29] Pham Dinh Khanh. “Image Segmentation”. In: <https://phamdinhkhanh.github.io/> (2020). URL: <https://bit.ly/369rdZA>.
- [30] Pham Dinh Khanh. “Pix2Pix GAN”. In: <https://phamdinhkhanh.github.io/> (2020). URL: <https://bit.ly/3umnGRo>.
- [31] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: [1412.6980 \[cs.LG\]](#).
- [32] A. Levin, D. Lischinski, and Y. Weiss. “Colorization using optimization”. In: *ACM transactions on graphics (TOG)*. Vol. 23. ACM. 2004, pp. 689–694.
- [33] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2018. arXiv: [1712.09913 \[cs.LG\]](#).
- [34] Q. Luan et al. “Natural image colorization”. In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association. 2007, pp. 309–320.
- [35] Eric Martin. “Reply: How do you decide to regularize between L1/L2 or best/greedy subset selection?” In: <https://www.quora.com/> (2014). URL: <https://bit.ly/2RIHhe>.

- [36] Richardson Santiago Teles de Menezes, Rafael Marrocos Magalhaes, and Helton Maia. “Object Recognition Using Convolutional Neural Networks”. In: (2019). DOI: [10.5772/intechopen.89726](https://doi.org/10.5772/intechopen.89726). URL: <https://bit.ly/3AswpWC>.
- [37] Luke Metz et al. *Unrolled Generative Adversarial Networks*. 2017. arXiv: [1611.02163 \[cs.LG\]](https://arxiv.org/abs/1611.02163).
- [38] Christopher Thomas BSc Hons. MIAP. “U-Nets with ResNet Encoders and cross connections”. In: <https://towardsdatascience.com/u-nets-with-resnet-encoders-and-cross-connections-5a2a2a2a2a2a> (2019). URL: <https://bit.ly/3ttEu8x>.
- [39] Divyanshu Mishra. “Transposed Convolution Demystified”. In: https://towardsdatascience.com/transposed-convolution-demystified-2020-10-10-1007978-3-319-94544-6_9 (2020). URL: <https://bit.ly/2TCybDE>.
- [40] Papia Nandi. “CNNs for Audio Classification”. In: <https://towardsdatascience.com/cnn-for-audio-classification-5a2a2a2a2a2a> (2021). URL: <https://bit.ly/3he5mGF>.
- [41] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. “Image Colorization Using Generative Adversarial Networks”. In: *Lecture Notes in Computer Science* (2018), pp. 85–94. ISSN: 1611-3349. DOI: [10.1007/978-3-319-94544-6_9](https://doi.org/10.1007/978-3-319-94544-6_9). URL: http://dx.doi.org/10.1007/978-3-319-94544-6_9.
- [42] Trung Thanh Nguyen. “Auto Colorize GrayScale Image With Deep Learning”. In: <https://forum.machinelearningcoban.com/> (2018). URL: <https://bit.ly/3eKdmwN>.
- [43] Tuan Nguyen. “Giới thiệu về xử lý ảnh”. In: <https://nttuan8.com/> (2019). URL: <https://bit.ly/3gYYqx7>.
- [44] Tuan Nguyen. “Pix2pix”. In: nttuan8.com/ (2020). URL: <https://bit.ly/3xGVR9h>.
- [45] Manisha Padala, Debojit Das, and Sujit Gujar. *Effect of Input Noise Dimension in GANs*. 2020. arXiv: [2004.06882 \[cs.LG\]](https://arxiv.org/abs/2004.06882).
- [46] Deepak Pathak et al. *Context Encoders: Feature Learning by Inpainting*. 2016. arXiv: [1604.07379 \[cs.CV\]](https://arxiv.org/abs/1604.07379).
- [47] Aladdin Persson. *Pix2Pix implementation from scratch*. YouTube. 2021. URL: <https://youtu.be/oLvmLJkmXuc>.
- [48] Phillip. “Reply: Random noise z augmentation with cGAN”. In: <https://github.com/> (2017). URL: <https://bit.ly/3oM7EPE>.
- [49] Reddit. “Introducing DeOldify: A Progressive, Self-Attention GAN based image colorization/restoration project”. In: <https://www.reddit.com/r/MachineLearning/> (2019). URL: <https://bit.ly/3jIfb0W>.
- [50] Reddit. “Reply: How are photos colorized? How long does it take?” In: <https://www.reddit.com/r/AskReddit/> (2012). URL: <https://bit.ly/3hBr729>.
- [51] Joseph Rocca. “Reply: Why are GAN used if GMN do the same thing.” In: <https://medium.com/> (2019). URL: <https://bit.ly/2UZJrut>.
- [52] Joseph Rocca. “Understanding Generative Adversarial Networks (GANs)”. In: <https://towardsdatascience.com/> (2019). URL: <https://bit.ly/3vzdPIU>.
- [53] Joseph Rocca. *Understanding Generative Adversarial Networks (GANs)*. YouTube. 2019. URL: <https://youtu.be/6v71JHFaZZ4>.
- [54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: [1505.04597 \[cs.CV\]](https://arxiv.org/abs/1505.04597).
- [55] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: [1609.04747 \[cs.LG\]](https://arxiv.org/abs/1609.04747).

- [56] Moein Shariatnia. “Colorizing black & white images with U-Net and conditional GAN - A Tutorial”. In: <https://towardsdatascience.com/> (2020). URL: <https://bit.ly/3eKaaBj>.
- [57] Nishank Singla. “UNet with ResBlock for Semantic Segmentation”. In: <https://medium.com/> (2019). URL: <https://bit.ly/3hhuufV>.
- [58] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014).
- [59] Tomasz Szandala. *Convolutional Neural Network for Blur Images Detection as an Alternative for Laplacian Method*. 2020. arXiv: [2010.07936 \[cs.CV\]](https://arxiv.org/abs/2010.07936).
- [60] Y.-W. Tai, J. Jia, and C.-K. Tang. “Local color transfer via probabilistic segmentation by expectation-maximization”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 747–754.
- [61] tejaskhot. “Reply: Optimization metric in Generative Adversarial Networks”. In: <https://stats.stackexchange.com/> (2017). URL: <https://bit.ly/3qvtfn1>.
- [62] Vũ Hữu Tiệp. “Tích chập hai chiều”. In: <https://machinelearningcoban.com/> (2018). URL: <https://bit.ly/2V5Me5e>.
- [63] Patricia Vitoria, Lara Raad, and Coloma Ballester. *ChromaGAN: Adversarial Picture Colorization with Semantic Class Distribution*. 2020. arXiv: [1907.09837 \[cs.CV\]](https://arxiv.org/abs/1907.09837).
- [64] Lifu Wang et al. *Is the Skip Connection Provable to Reform the Neural Network Loss Landscape?* 2020. arXiv: [2006.05939 \[cs.LG\]](https://arxiv.org/abs/2006.05939).
- [65] T. Welsh, M. Ashikhmin, and K. Mueller. “Transferring color to greyscale images”. In: *ACM Transactions on Graphics (TOG)*. Vol. 21. ACM. 2002, pp. 277–280.
- [66] Wikipedia. “AlexNet”. In: <https://en.wikipedia.org/wiki/> (2021). URL: <https://en.wikipedia.org/wiki/AlexNet>.
- [67] Wikipedia. “Binary image”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Binary_image.
- [68] Wikipedia. “CIELAB color space”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/CIELAB_color_space.
- [69] Wikipedia. “CMYK color model”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/CMYK_color_model.
- [70] Wikipedia. “Color space”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Color_space.
- [71] Wikipedia. “Convolution”. In: <https://en.wikipedia.org/wiki/> (2021). URL: <https://en.wikipedia.org/wiki/Convolution>.
- [72] Wikipedia. “Cross entropy”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Cross_entropy.
- [73] Wikipedia. “Deep learning”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Deep_learning.
- [74] Wikipedia. “Dot product”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Dot_product.
- [75] Wikipedia. “Grayscale”. In: <https://en.wikipedia.org/wiki/> (2021). URL: <https://en.wikipedia.org/wiki/Grayscale>.
- [76] Wikipedia. “HSL and HSV”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/HSL_and_HSV.

- [77] Wikipedia. “Kernel (image processing)”. In: <https://en.wikipedia.org/wiki/> (2021). URL: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)).
- [78] Wikipedia. “Machine learning”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Machine_learning.
- [79] Wikipedia. “Max-pooling / Pooling”. In: <https://computersciencewiki.org/> (2021). URL: https://computersciencewiki.org/index.php/Max-pooling_-_Pooling.
- [80] Wikipedia. “Mean squared error”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Mean_squared_error.
- [81] Wikipedia. “RGB color model”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/RGB_color_model.
- [82] Wikipedia. “RGBA color model”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/RGBA_color_model.
- [83] Wikipedia. “Root-mean-square deviation”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Root-mean-square_deviation.
- [84] Wikipedia. “Vanishing gradient problem”. In: <https://en.wikipedia.org/wiki/> (2021). URL: https://en.wikipedia.org/wiki/Vanishing_gradient_problem.
- [85] Fisher Yu and Vladlen Koltun. *Multi-Scale Context Aggregation by Dilated Convolutions*. 2016. arXiv: [1511.07122 \[cs.CV\]](https://arxiv.org/abs/1511.07122).
- [86] Aston Zhang et al. “Padding and Stride”. In: *Dive into Deep Learning* 6 (2019).
- [87] R. Zhang, P. Isola, and A. A. Efros. “Colorful image colorization”. In: *European conference on computer vision*. Springer. 2016, pp. 649–666.
- [88] Rongyu Zhang et al. “Comparison of Backbones for Semantic Segmentation Network”. In: *Journal of Physics: Conference Series* 1544 (2020). DOI: [10.1088/1742-6596/1544/1/012196](https://doi.org/10.1088/1742-6596/1544/1/012196). URL: <https://bit.ly/3hVPOrY>.
- [89] Jun-Yan Zhu. *CycleGAN and pix2pix in PyTorch*. 2020. URL: <https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>.