

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.

Em cũng muốn gửi lời cảm ơn chân thành đến trường Đại học Bách Khoa TP.HCM, khoa Điện–Điện Tử, bộ môn Viễn Thông đã tạo mọi điều kiện để em có thể hoàn thành tốt đồ án của mình.

Cảm ơn các anh chị khoá trên cùng các bạn cùng khoa và một số 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 nghiên cứu của đồ án.

Em xin chân thành cảm ơn.

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

Sinh Viên



# Tóm tắt đồ án

Tạo màu cho ảnh xám là một chủ đề nóng hổi 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. 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, một 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  $70 \times 70$ , được triển khai và huấn luyện với tập dữ liệu 10,000 bức ảnh của tập dữ liệu COCO để xử lí vấn đề này. Chất lượng mô hình thông qua một cuộc khảo sát thực tế 100 người được đánh giá ở mức trung bình-khá.

Bên cạnh đó, xây dựng được 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

<b>1 Giới thiệu</b>	<b>1</b>
1.1 Tổng quan . . . . .	1
1.2 Mục tiêu . . . . .	1
<b>2 Lý thuyết</b>	<b>2</b>
2.1 Không gian màu . . . . .	2
2.2 Những phương pháp được đề xuất . . . . .	3
2.3 Mạng đối nghịch tạo sinh (GAN - Generative Adversarial Network) . . . . .	4
2.4 Mạng Unet . . . . .	7
2.5 Kiến trúc được áp dụng . . . . .	9
2.6 Học chuyển giao cho bộ sinh bằng xương sống ResNet18 . . . . .	12
<b>3 Triển khai mô hình</b>	<b>12</b>
3.1 Ngôn ngữ lập trình và thư viện chính . . . . .	12
3.2 Tập dữ liệu . . . . .	13
3.3 Chuẩn hoá và làm giàu dữ liệu . . . . .	13
3.4 Huấn luyện mô hình . . . . .	14
<b>4 Thủ nghiệm và đánh giá mô hình</b>	<b>15</b>
4.1 Mô hình GAN với bộ sinh đơn giản . . . . .	15
4.2 Mô hình GAN với bộ sinh có xương sống ResNet18 . . . . .	15
4.3 So sánh bộ sinh có xương sống ResNet18 trước và sau khi huấn luyện đối nghịch	17
4.4 Khảo sát đánh giá chất lượng mô hình . . . . .	18
4.5 Kết quả sau khi huấn luyện lại với tập dữ liệu lớn hơn . . . . .	22
<b>5 Ứng dụng</b>	<b>23</b>
5.1 Giao diện lập trình ứng dụng (API - Application Programming Interface) bằng khung phần mềm Django của Python . . . . .	23
5.2 Ứng dụng web với khung phần mềm VueJS của Javascript và API . . . . .	24
<b>6 Tổng kết</b>	<b>25</b>
6.1 Kết luận . . . . .	25
6.2 Hướng cải thiện mô hình . . . . .	25
6.3 Hướng phát triển . . . . .	26
<b>7 Phục lục - Ảnh màu</b>	<b>27</b>

## Danh sách hình

1.1	Nải chuối có thể màu lục hoặc màu vàng. . . . .	1
2.1	Các kênh màu đỏ, lục và lam được tách ra riêng biệt [31]. . . . .	2
2.2	Không gian màu Lab. . . . .	2
2.3	Các giá trị <b>L</b> , <b>a</b> và <b>b</b> được tách ra riêng biệt [31]. . . . .	3
2.4	Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh [25]. . . . .	3
2.5	Mô tả ý tưởng thuật toán ảnh có bối cảnh tương tự [25]. . . . .	4
2.6	Ảnh mặt người sinh ra bởi StyleGAN [16]. . . . .	5
2.7	Minh họa bộ sinh và bộ phân biệt trong mạng GAN [8]. . . . .	6
2.8	Kiến trúc mạng đối nghịch tạo sinh. . . . .	6
2.9	Mô phỏng quá trình huấn luyện mạng GAN [29]. . . . .	7
2.10	Kiến trúc mô hình mạng Unet [30]. . . . .	8
2.11	Mô hình GAN dự đoán ảnh dựa vào đường viền. Một bài toán tương tự với bài toán tạo màu cho ảnh [15]. . . . .	9
2.12	Hình ảnh giả sử được đầu vào chia thành $k \times k$ Patch [18]. . . . .	10
2.13	Mô tả vùng nhận thức qua các tầng. . . . .	11
3.1	Hình ảnh trích từ tập dữ liệu COCO. . . . .	13
3.2	Giá trị các thành phần mắt mát của bộ phân biệt và bộ sinh qua từng epoch. . . . .	14
4.1	Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản sau epoch thứ 34 trên tập dữ liệu COCO. . . . .	15
4.2	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). . . . .	16
4.3	Mô hình hoàn toàn bế tắc trước logo kỷ niệm 60 năm của trường ĐH Bách Khoa TP.HCM. . . . .	16
4.4	Trường hợp đặc biệt khi mô hình làm việc tệ với ảnh có đặc trưng đơn giản. . . . .	16
4.5	Những tấm ảnh đen trắng được lựa chọn để so sánh sự khác biệt. . . . .	17
4.6	Phía trên và dưới lần lượt là kết quả của bộ sinh trước và sau khi huấn luyện đối nghịch. . . . .	17
4.7	Một số kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc. . . . .	18
4.8	Thêm Một vài kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc. . . . .	19
4.9	Phản mô tả của bài đánh giá. . . . .	19
4.10	Phản câu hỏi và số lượng người tham gia cuộc khảo sát thực hiện thông qua Google Forms. . . . .	20
4.11	Những ảnh có điểm trung bình cao nhất. . . . .	22
4.12	Những ảnh có điểm trung bình thấp nhất. . . . .	22
4.13	So sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại. . . . .	22
4.14	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. . . . .	23
5.1	Bảo mật CORS cản trở khi thử nghiệm API . . . . .	24
5.2	Phản hồi trả về của endpoint . . . . .	24
5.3	Thử nghiệm API trên phần mềm Postman . . . . .	24
5.4	Ứng dụng web tạo màu đơn giản bằng VueJS kết hợp API . . . . .	25
6.1	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) . . . . .	26



## Danh sách bảng

4.1	Kết quả bình chọn cho các ảnh.	21
4.2	Điểm trung bình của các ảnh.	21



*Trang này được cố ý để trống.*

# 1 Giới thiệu

## 1.1 Tổng quan

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

Hiện nay, hầu hết những người thợ chỉnh sửa ảnh đều sử dụng phương pháp thủ công. Tuy có sự giúp đỡ của các công cụ hỗ trợ như các phần mềm chỉnh sửa ảnh trên máy tính, để có thể có một tấm ảnh đẹp cũng đòi hỏi mất nhiều thời gian, 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 đổi 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. Do đó, việc khôi phục lại đúng chính xác màu cho một bức ảnh bất kì là chuyện gần như bất khả thi và không cần thiết. Nhưng cũng không vì thế mà ta có thể gán một màu bất kì cho một đổi tượng nào đó, giả sử như việc màu da người là màu xanh biển hoặc mặt trời là màu xanh lá. Việc 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, hiển nhiên không phải là mục đích để chúng ta khôi phục.



Hình 1.1: Nải chuối có thể màu lục hoặc màu vàng.

Trước khi có sự xuất hiện của của ngành học sâu 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. 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 bùn nổ trong ngành thị giác máy tính, rất nhiều mô hình học sâu ứng dụng mang tính chất đã có thể học và tạo ra màu cho những ảnh xám.

## 1.2 Mục tiêu

Đồ án được đưa ra với mục đích xây dựng và huấn luyện một mô hình học sâu để tạo màu cho ảnh xám. Yêu cầu của màu tạo ra không cần thiết phải giống hoàn toàn so với màu thực, thay vào đó độ chân thực, hợp lí sẽ là thứ được hướng đến.

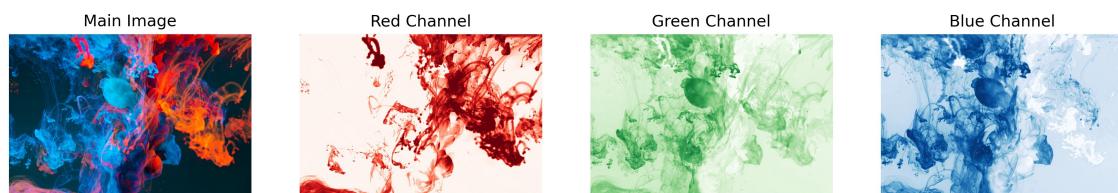
Những nội dung chính được nghiên cứu, trình bày trong đồ án bao gồm:

- Lựa chọn không gian màu thích hợp để sử dụng cho bài toán tạo màu.
- Mô hình học sâu để tạo màu - mạng đối nghịch tạo sinh với bộ kiến trúc mạng Unet.
- Xây dựng kiến trúc mạng Unet dựa vào xương sống mạng ResNet bằng học chuyển giao.
- Triển khai mô hình sử dụng ngôn ngữ lập trình Python.

## 2 Lý thuyết

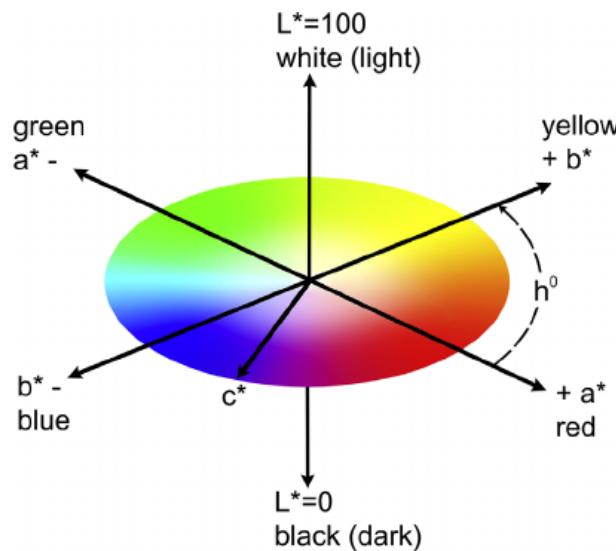
### 2.1 Không gian màu

Hầu hết khi làm việc với ảnh số, ta sẽ thao tác với ảnh có không gian màu là **RGB**, tức ảnh có 3 kênh màu đỏ-lục-lam. Đây cũng là không gian màu phổ biến với đại đa số mọi người. Với ảnh thuộc không gian màu **RGB**, mỗi điểm ảnh sẽ có 3 giá trị **R-G-B**, mỗi giá trị nằm trong đoạn [0, 255], tương ứng với 3 kênh màu để tạo nên được màu của chính điểm ảnh.



Hình 2.1: Các kênh màu đỏ, lục và lam được tách ra riêng biệt [31].

Ngoài RGB, một không gian màu khác cũng được sử dụng khá nhiều là không gian màu **L\*a\*b\***, không gian này cũng quan tâm đến 3 thông số của mỗi điểm ảnh. Thông số đầu tiên là **L\*** đại diện cho độ sáng 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-dỏ và vàng-lam. Giá trị **a\*** càng thấp thì điểm ảnh nghiêng về lục nhiều, dỏ ít. Ngược lại thì lục ít, dỏ nhiều và tương tự với giá trị **b\***.

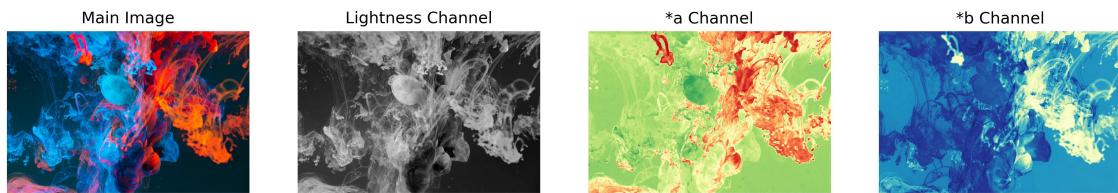


Hình 2.2: Không gian màu Lab.

Khoảng giá trị của kênh **L\*** nằm trong đoạn [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 đoạn [-128, 127].

Giữa hai không gian màu trên, 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** [35]:

- i) Không gian màu **L\*a\*b\*** được thiết kế để đồng nhất về cảm giác, điều này làm cho không gian màu này trở nên lý tưởng cho việc xử lí máy tính.

Hình 2.3: Các giá trị **L**, **a** và **b** được tách ra riêng biệt [31].

- 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**.

## 2.2 Những phương pháp được đề xuất

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: Dựa vào những màu ban được đánh dấu ban đầu, dựa vào mẫu và cuối cùng là dùng 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 là 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.

Cách làm đơn giản nhất trong phương pháp **dựa vào những màu được đánh dấu ban đầu** có thể kể đến là **đánh dấu màu một vài điểm ảnh và dùng thuật toán loang**, được đề xuất bởi Levin [20]. 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 [13]. Hay theo như Luan [21], ta đánh dấu mỗi màu cho mỗi phân đoạn khác nhau.



Hình 2.4: Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh [25].

Những phương pháp thuộc loại phương pháp trên 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.

**Dựa vào mẫu** là một loại phương pháp sử dụng một mẫu giống hoặc tương tự để giải quyết trọn vẹn bài toán. Với bài toán tô màu, phương pháp được khởi xướng bởi Welsh [34] sẽ **dựa**

vào màu của ảnh có bối cục tương tự. Ta chọn ra một tấm ảnh mẫu có bối cục 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 [14] đề xuất việc đưa màu của những phân đoạn của ảnh mẫu sang để dùng để đánh dấu màu rồi dùng thuật toán loang. Một hướng khác, Tai [32] 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 xứng.



Hình 2.5: Mô tả ý tưởng thuật toán ảnh có bối cục tương tự [25].

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 cục tương tự so với ảnh cần xử lý.

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 [7], 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 nhất trong phương pháp loại này có thể nói đến là Zhang [36], bằng việc học phân bố 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.

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 ứng dụng khung mô hình mạng học sâu **pix2pix** [15] được đề xuất bởi Isola, một mô hình sử dụng mạng **GAN** [9] có điều kiện với bộ sinh có kiến trúc mạng **Unet** [30] kết hợp với bộ phân biệt Patch. Vì thế, để có thể có cái nhìn rõ ràng về kiến trúc mô hình được áp dụng xử lí bài toán tạo màu, ta cần sẽ đi qua những thành phần mạng chính tạo nên mô hình là GAN và Unet.

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

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 (xem hình 2.6) là do GAN sinh ra, không phải mặt người thật.

G - Generative ý chỉ sinh, N - Network là mạng, còn A - Adversarial là đối nghịch. Lí 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



Hình 2.6: Ảnh mặt người sinh ra bởi StyleGAN [16].

biết 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, 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.

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.

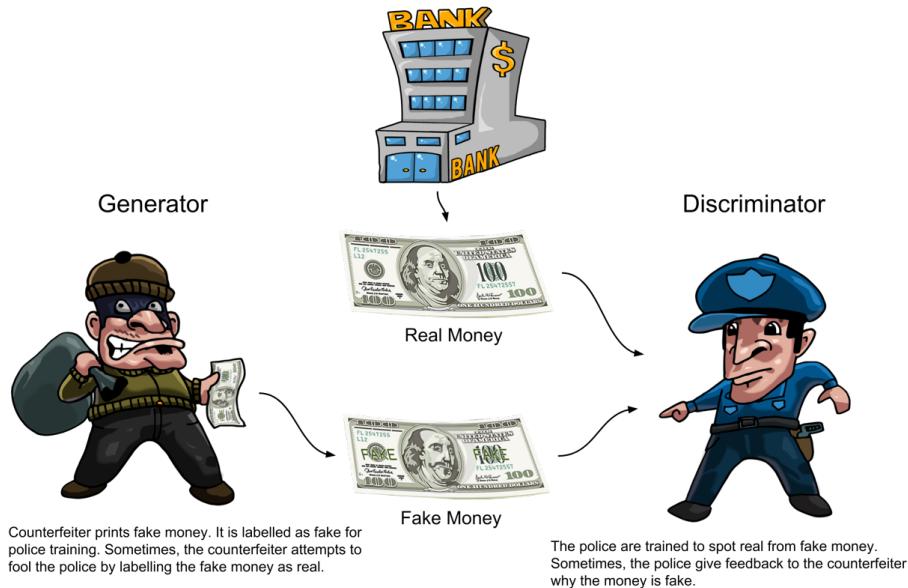
Ý tưởng của GAN bắt nguồn từ [Non-Zero-Sum Games](#)<sup>1</sup>, là một trò chơi đối kháng giữa 2 người, nếu một trong hai người thắng, thì người còn lại sẽ thua. Ở mỗi lượt, thì cả 2 đều muốn tối đa hóa cơ hội thắng của mình và tối thiểu hóa cơ hội thắng của đối phương. Trong lý thuyết trò chơi thì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

Nhìn theo khía cạnh kỹ thuật, 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  $X = \{x_1, x_2, \dots, x_n\}$  và  $X' = \{x'_1, x'_2, \dots, x'_n\}$  có được rút ra từ cùng một phân phối hay không. Sự khác biệt chính giữa hầu hết những bài nghiên cứu thông 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**<sup>3</sup> để cung cấp tín hiệu cho việc huấn luyện cho bộ

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

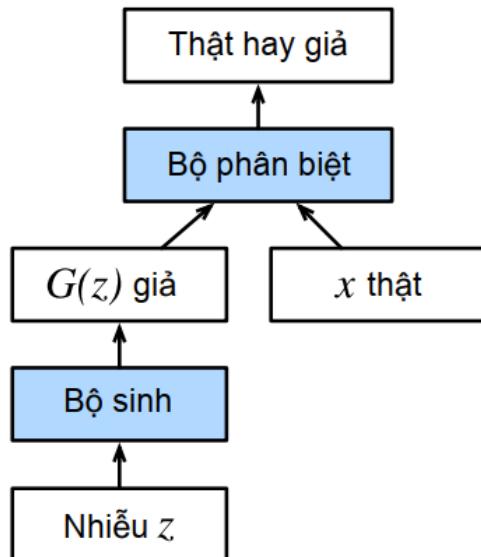
<sup>2</sup>Jørgen Veisdal, ‘The Nash equilibrium, explained’, <https://bit.ly/3ea57Lr>

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



Hình 2.7: Minh họa bộ sinh và bộ phân biệt trong mạng GAN [8].

sinh. Điều này cho phép ta cải thiện bộ sinh để có thể sinh dữ liệu tối kỵ 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 2.8: Kiến trúc mạng đối nghịch tạo sinh.

Bộ phân biệt là một bộ phân loại nhị phân nhằm phân biệt đầu vào  $\mathbf{x}$  là thật (từ dữ liệu thật) hoặc giả (từ bộ sinh), 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  $S(x) = 1/(1 + e^{-x})$ ,  $R = (0, 1)$  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. Hàm mất mát của bộ phân biệt cũng vì thế mà có dạng entropy chéo, nghĩa là:

$$\min_D \{-y \log D(\mathbf{x}) - (1 - y) \log (1 - D(\mathbf{x}))\}$$

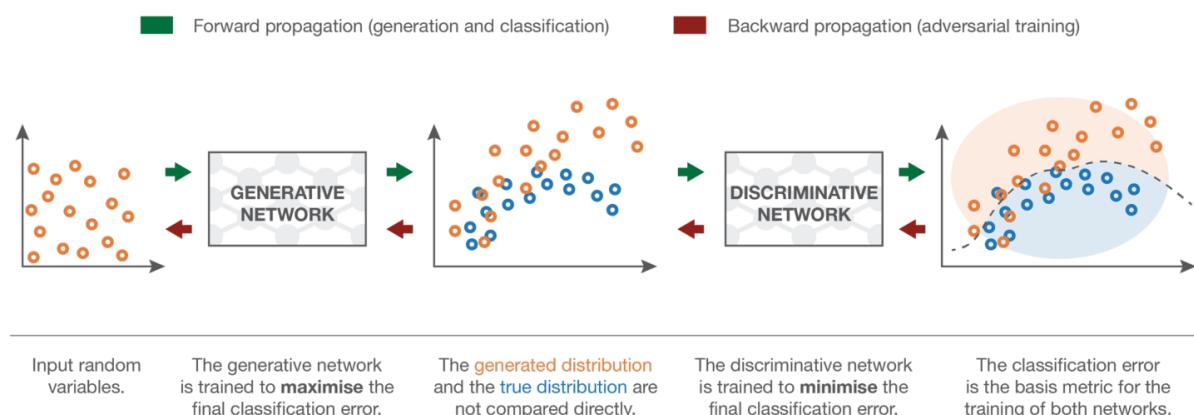
Còn đối với bộ sinh, sẽ được học theo kiểu không giám sát. Trước tiên nó cần được cho vài tham số ngẫu nhiên được xem là nhiễu  $\mathbf{z} \in \mathbb{R}^d$  (**số chiều d thường là 100<sup>4</sup>**) từ một nguồn, ví dụ phân phối chuẩn  $\mathbf{z} \sim \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 1)$ . Mục tiêu của bộ sinh là đánh lừa bộ phân biệt để phân loại  $\mathbf{x}' = G(\mathbf{z})$  là dữ liệu thật, nghĩa là, ta muốn  $D(G(\mathbf{z})) \approx 1$ . Nói cách 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 khi  $y = 0$ , tức là:

$$\max_G \{-\log(1 - D(G(\mathbf{z})))\}$$

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

$$\begin{aligned} & \min_D \max_G \{-\mathbb{E}_{\mathbf{x} \sim \text{data}} \log D(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim \text{noise}} \log(1 - D(G(\mathbf{z})))\} \\ \Leftrightarrow & \min_G \max_D \{\mathbb{E}_{\mathbf{x} \sim \text{data}} \log D(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim \text{noise}} \log(1 - D(G(\mathbf{z})))\} \end{aligned}$$

Kí hiệu  $\mathbb{E}$  ở đây chính là kì vọng toán, tương đương việc lấy trung bình của tất cả dữ liệu.



Hình 2.9: Mô phỏng quá trình huấn luyện mạng GAN [29].

Quá trình huấn luyện mạng GAN cũng không khác gì so với những mạng học sâu khác là bao. Như được mô tả trong hình 2.9, quá trình được bắt đầu bằng việc lan truyền thuận khi bộ sinh lấy một biến ngẫu nhiên  $\mathbf{z}$  từ phân bố được chọn ban đầu rồi dựa vào đó sinh ra một biến ngẫu nhiên  $G(\mathbf{z})$  thuộc một phân bố khác. Dữ liệu thật  $\mathbf{x}$  hiển nhiên sẽ có phân bố khác với phân bố của  $G(\mathbf{z})$ . Bộ phân biệt sẽ có nhiệm vụ tạo ra đường biên thích hợp để phân biệt hai phân bố  $G(\mathbf{z})$  và  $\mathbf{x}$ . Kết thúc quá trình lan truyền thuận là quá trình lan truyền ngược, bộ phân biệt được huấn luyện tốt hơn để phân biệt hai phân bố đồng thời giúp cho bộ sinh điều chỉnh để có thể sinh ra  $G(\mathbf{z})$  có phân bố giống với lại phân bố của  $\mathbf{x}$ . Ta chỉ điều chỉnh lại phân bố của  $G(\mathbf{z})$  chứ không điều chỉnh phân bố của  $\mathbf{z}$  vì không cần thiết và phức tạp.

## 2.4 Mạng Unet

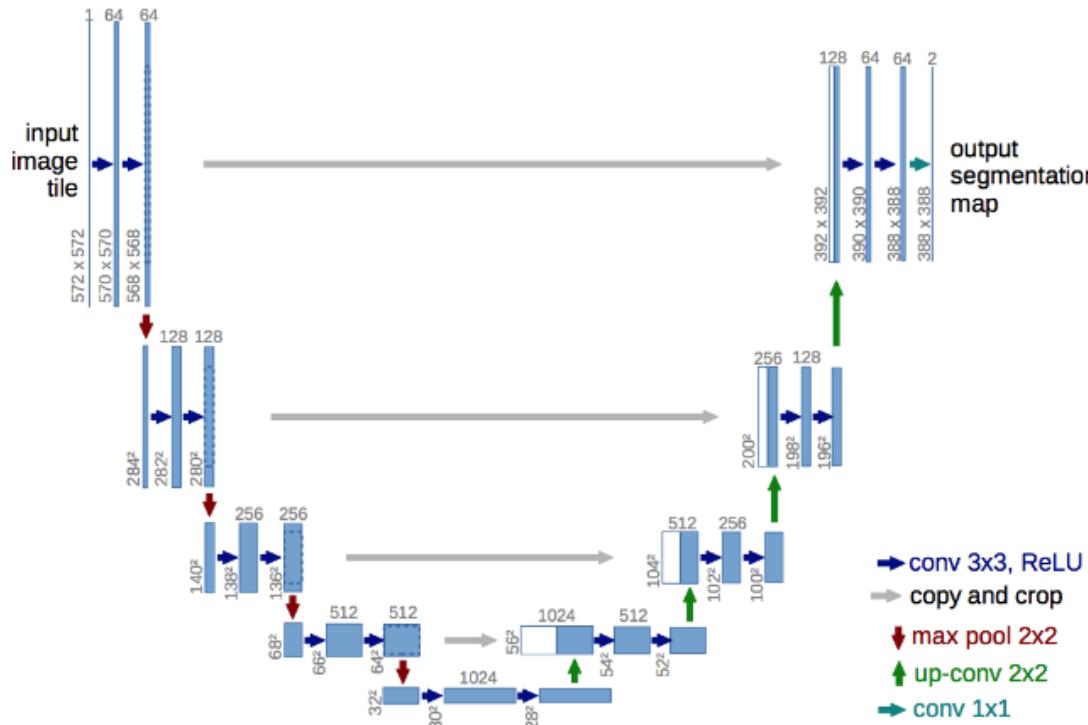
Unet là một kiến trúc được phát triển nhằm phân vùng các cấu trúc nơ-ron thần kinh trong não người. Kiến trúc này lần đầu được áp dụng đã giành chiến thắng trong cuộc thi EM segmentation challenge at ISBI 2012.<sup>5</sup>

<sup>4</sup>Reddit, “Why is Z-dimension for GANs usually 100?”, <https://bit.ly/33nXNpk>

<sup>5</sup>ISBS Challenge: Segmentation of neuronal structures in EM stacks, <https://bit.ly/2RAIyqR>

Mạng Unet bao gồm 2 nhánh đối xứng nhau hình chữ U nên được gọi là Unet. Kiến trúc mạng bao gồm 2 phần là **thu hẹp** ở nhánh trái và **phần mở rộng** ở nhánh 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 (*WHAT*). Vai trò của phần thu hẹp tương tự như một bộ mã hoá. Một mạng tích chập sâu sẽ đóng vai trò trích lọc đặc trưng. Lý do nhánh được gọi là thu hẹp vì kích thước dài và rộng của các tầng giảm dần. Độ phân giải được giảm bằng cách sử dụng **gộp cực đại** 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 lai để biết được vị trí (*WHERE*) 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 hoặc một phương thức khác là sử dụng **tích chập giãn nở**. Nhưng được áp dụng nhiều nhất là **tích chập chuyển vị** cùng với đệm và sải bước thích hợp.



Hình 2.10: Kiến trúc mô hình mạng Unet [30].

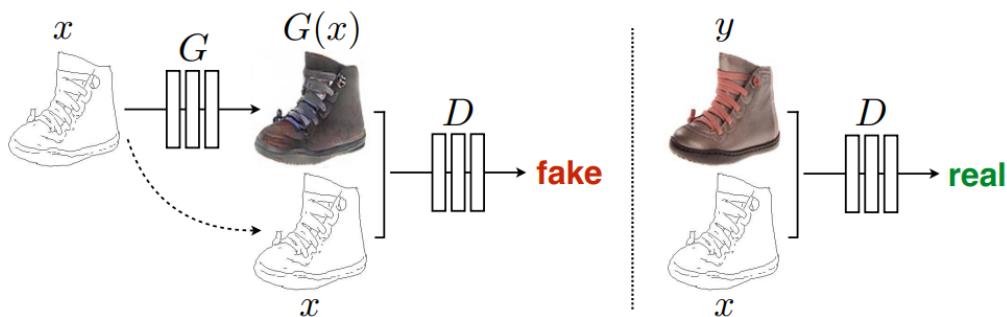
Đặc trưng riêng trong cấu trúc của Unet đó là có những **kết nối tắt** đối xứng giữa các tầng của nhánh bên trái tương ứng với các tầng bên nhánh bên phải. Việc kết nối tắt theo kiến trúc Unet là một điểm mới so với kiến trúc mã hoá và giải mã thông thường nhằm bổ sung thêm thông tin và gia tăng độ chính xác. Chi tiết về kết nối tắt, có thể tham khảo một số mạng tích chập hiện đại sử dụng như phương pháp trên [11], [12]. Ngoài ra kết nối tắt còn khắc phục hiện tượng gradient biến mất [1].

## 2.5 Kiến trúc được áp dụng

Bài toán tạo màu có thể được tóm tắt như sau:

Cho một ảnh đầu vào có kích thước  $S \times S = 256 \times 256$  chỉ có thông tin cường độ mức sáng mỗi điểm ảnh là  $\mathbf{L}^* \in \mathbb{R}$ , ta cần phải xây dựng một ảnh xạ  $\mathcal{G} : \mathbf{L}^* \rightarrow (\widehat{\mathbf{a}}^*, \widehat{\mathbf{b}}^*) \in \mathbb{R}^2$  sao cho  $\mathcal{I} = \left\{ p_{ij} = \left( \mathbf{L}^*_{ij}, \widehat{\mathbf{a}}^*_{ij}, \widehat{\mathbf{b}}^*_{ij} \right) | p_{ij} \text{ là điểm ảnh ở hàng } i, \text{ cột } j \right\}$  là một ảnh màu hợp lí, chân thực với  $\widehat{\mathbf{a}}^*$  và  $\widehat{\mathbf{b}}^*$  lần lượt là giá trị 2 kênh màu lục-đỏ và vàng-lam trong không gian màu  $\mathbf{L}^* \mathbf{a}^* \mathbf{b}^*$ .

Trong mạng GAN được huấn luyện, bộ sinh  $G$  sẽ sinh ra 2 kênh màu  $\mathbf{a}^*$  và  $\mathbf{b}^*$  - Chính là ánh xạ  $\mathcal{G}$  cần tìm. Còn bộ phân biệt  $D$  có nhiệm vụ phân biệt ảnh màu đưa vào là thật hay giả.



Hình 2.11: Mô hình GAN dự đoán ảnh dựa vào đường viền. Một bài toán tương tự với bài toán tạo màu cho ảnh [15].

Bộ sinh theo kiến trúc mạng Unet với đầu vào là một ảnh chỉ có một kênh biểu thị cho cường độ sáng mỗi điểm ảnh là  $\mathbf{X} \in \mathbb{R}^{S \times S \times 1}$ . Bức ảnh sẽ được truyền vào một kiến trúc mạng tích chập để trích lọc đặc trưng thông qua quá trình mã hoá. 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 hoá 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  $\widehat{\mathbf{Y}} \in \mathbb{R}^{S \times S \times 2}$ . Khi đó, ảnh giả được bộ sinh tạo ra là:

$$\widehat{\mathcal{I}} = \left\{ p_{ij} = \left( \mathbf{L}^*_{ij}, \widehat{\mathbf{a}}^*_{ij}, \widehat{\mathbf{b}}^*_{ij} \right) | \mathbf{L}^*_{ij} \in \mathbf{X}, \left( \widehat{\mathbf{a}}^*_{ij}, \widehat{\mathbf{b}}^*_{ij} \right) \in \widehat{\mathbf{Y}} \right\}$$

Đầu vào của bộ phân biệt là ảnh  $\mathcal{I} = (\mathbf{X}, \mathbf{Y}) \in \mathbb{R}^{S \times S \times 3}$  hoặc ảnh  $\widehat{\mathcal{I}} = (\mathbf{X}, \widehat{\mathbf{Y}}) \in \mathbb{R}^{S \times S \times 3}$ . Các ảnh sẽ có nhãn đầu ra là thật nếu cặp dữ liệu đầu vào có  $\mathbf{Y}$  được lấy từ tập huấn luyện và trái lại là khi dữ liệu đầu vào có  $\widehat{\mathbf{Y}}$  được sinh ra từ bộ sinh thì sẽ mang nhãn giả. Một điểm khác biệt dễ thấy so với bộ phân biệt ở mạng GAN thông thường là thay vì chỉ nhận được đầu vào là  $\mathbf{Y}$  hoặc  $\widehat{\mathbf{Y}}$  cho việc phân loại mà còn kèm theo  $\mathbf{X}$ . Đây chính là điều kiện ràng buộc cho quyết định của bộ sinh, vì muốn biết màu thật hay giả thì phải biết màu đó của thứ gì.

Hàm mất mát của mô hình GAN của chúng ta cũng không quá khác biệt so với hàm mất mát của một mạng GAN thông thường, cụ thể là:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{\mathbf{X}, \mathbf{Y}} \log D(\mathbf{X}, \mathbf{Y}) + \mathbb{E}_{\mathbf{X}} \log (1 - D(\mathbf{X}, G(\mathbf{X})))$$

Nhiều  $\mathbf{z}$  không được đề cập trong công thức hàm mất mát ở đây, tuy nhiên trong [15] vẫn được xuất hiện. Các tác giả đã cố gắng tạo ra nhiều từ việc sử dụng kỹ thuật **dropout** trong giai

đoạn huấn luyện và cả thử nghiệm. Nếu ta không dùng kỹ thuật dropout, mô hình vẫn sẽ có thể được huấn luyện và đưa ra kết quả dự đoán tốt tuy nhiên không được tổng quát, kết quả dự đoán của mô hình sẽ khá bị bó buộc khi phân phối sẽ chỉ có thể là hàm delta. Nhưng theo như trả lời của tác giả [28], mô hình không thực sự cần đến nhiều nếu đầu vào đủ phức tạp. Lúc đó đầu vào đóng vai trò như nhiễu. Và với bài toán tô màu, đầu vào là một bức ảnh xám được đánh giá là đủ phức tạp nên thành phần nhiễu được phép bỏ qua.

Trong [15] cũng chỉ ra rằng, bằng việc kết hợp giữa hàm mất mát của GAN với một hàm mất mát truyền thống như chuẩn bậc 1 hoặc chuẩn bậc 2 cho bộ sinh thì khi đó, ngoài việc đánh lừa được bộ phân biệt thì nó còn phải làm sao sinh ảnh giống với kết quả có thực trong dữ liệu huấn luyện thông qua tối thiểu hóa chuẩn bậc 1 hoặc chuẩn bậc 2. Bên dưới là thành phần chuẩn bậc 1 được lựa chọn để giải quyết bài toán tạo màu:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{\mathbf{X}, \mathbf{Y}} \|\mathbf{Y} - G(\mathbf{X})\|_1$$

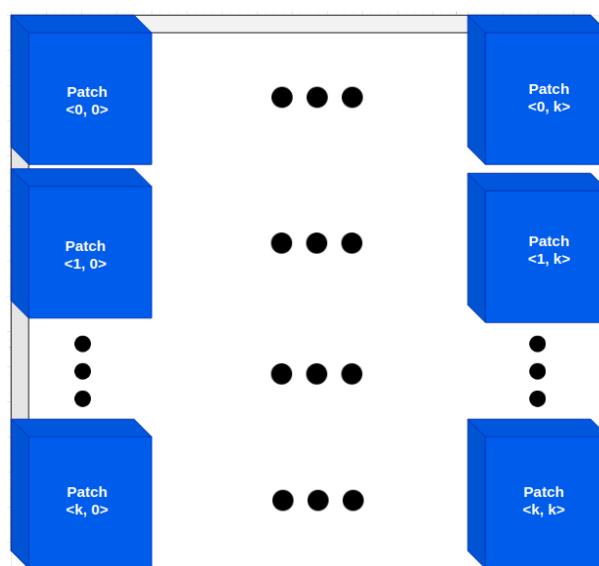
Lí do để chọn chuẩn bậc 1 thay vì chuẩn bậc 2 có thể được lý giải một cách trực quan, là do chuẩn bậc 2 có xu hướng trừng phạt mất mát nhiều hơn so với chuẩn bậc 1, do đó điều này làm bối rối kết quả dự đoán của bộ sinh khiến cho mô hình có xu thế bảo thủ, sẽ đưa dự đoán bằng cách lấy giá trị trung bình 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 dự đoán phiêu lưu hơn. Để hiểu hơn về việc lựa chọn chuẩn bậc 1 hay chuẩn bậc 2, có thể tham khảo [22].

Cuối cùng, mô hình ta cần phải tối ưu là:

$$G^* = \arg \min_{G} \max_{D} \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

Trong đó,  $\lambda$  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  $\lambda$  sẽ là một số lớn hơn 1, và thường giá trị được chọn sẽ là  $\lambda = 100$ .



Hình 2.12: Hình ảnh giả sử được đầu vào chia thành  $k \times k$  Patch [18].

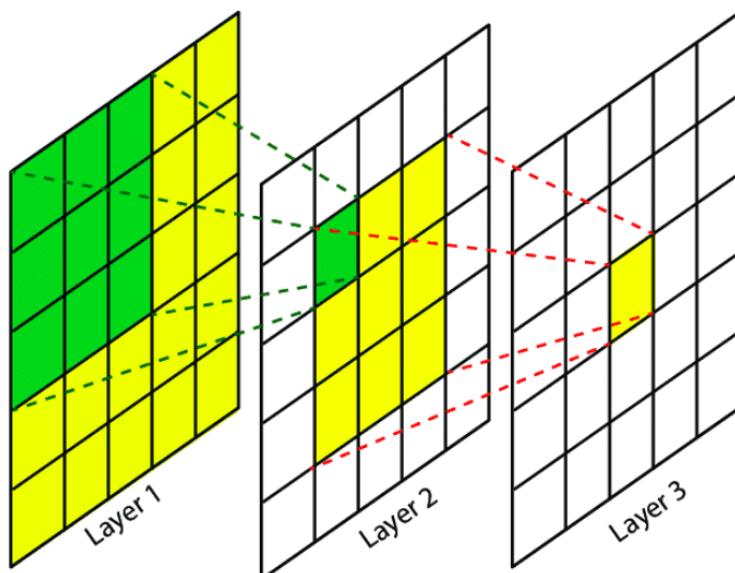
Một điểm khác biệt nữa của bộ phân biệt trong mô hình này so với những bộ phân biệt trong các mô hình GAN thông thường là bộ phân biệt của chúng ta là làm việc theo Patch. Một bộ phân biệt thông thường chỉ trả về một số vô hướng thì bộ phân biệt Patch sẽ trả về một ma

trận  $\mathbf{P}_D \in \mathbb{R}^{k \times k}$  bằng cách phân loại từng vùng  $N \times N$  của đầu vào là thật hay giả thay vì toàn bộ một lúc. Những vùng  $N \times N$  này được gọi là vùng nhận thức và phân loại mỗi vùng nhận thức sẽ cho một giá trị  $[\mathbf{P}_D]_{ij}$ . Về bản chất, bộ phân biệt Patch vẫn là một kiến trúc mang tính 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. Mà thay vào 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 [15] là  $70 \times 70$ .

Kiến trúc các tầng của một bộ phân biệt Patch  $70 \times 70$  có thể biểu diễn đơn giản như sau:

$$I \rightarrow 2C64 \rightarrow 2C128 \rightarrow 2C256 \rightarrow 1C512 \rightarrow O$$

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  $\mathbf{P}_D$  và kích hoạt hàm sigmoid để lấy xác suất. Tất cả các tầng đều sử dụng kích thước bộ lọc là 4 cùng với kích thước đệm là 1.



Hình 2.13: 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. Ta đã biết rằng, kích thước đầu ra sau khi qua một bộ lọc được tính bằng [3]:

$$W_{\text{output}} = \left\lceil \frac{W_{\text{input}} - K + 2P}{S} \right\rceil + 1$$

Như vậy, bằng các phép biến đổi ta suy ra được:

$$W_{\text{input}} = K - 2P + S(W_{\text{output}} - 1)$$

Tuy nhiên, khi suy ngược lại để tính vùng nhận thức ta cần quan tâm vùng ảnh không có đệm. Do đó công thức để tính kích thước vùng nhận thức  $F$  từng tầng sẽ là:

$$F_{\text{input}} = K + S(F_{\text{output}} - 1)$$

Áp dụng công thức trên, ta sẽ tính được kích thước vùng nhận thức của mỗi giá trị đầu ra theo như kết quả sau đây:

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

## 2.6 Học chuyển giao cho bộ sinh bằng xương sống ResNet18

Trong mạng cGAN của chúng ta, 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 bộ sinh được đề xuất trong [15] như sau:

nhánh trái:  $I \rightarrow C64 \rightarrow C128 \rightarrow C256 \rightarrow C512 \rightarrow C512 \rightarrow C512 \rightarrow C512$   
nhánh phải:  $\rightarrow D512 \rightarrow D1024 \rightarrow D1024 \rightarrow D1024 \rightarrow D512 \rightarrow D256 \rightarrow D128 \rightarrow D2$

Trong đó,  $Ck$  là một tầng tích chập  $k$  bộ lọc,  $Dk$  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 hạn khá thử thách [4.1].

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 [23]. Có 3 họ mạng được chủ yếu dùng để làm xương sống cho Unet là VGG, ResNet và Xception [37]. Trong những họ mạng trên, ResNet18 có kích thước không quá lớn [17], phù hợp để huấn luyện với phần cứng bị giới hạn. Một lưu ý là mạng ResNet18 sẽ chỉ làm phẳng mã hoá, tức nhánh bên trái của chữ U chứ không hề xây dựng toàn bộ cả mạng Unet. Phần mã hoá sẽ được tự động khởi tạo bởi thư viện fastai<sup>6</sup>.

## 3 Triển khai mô hình

### 3.1 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 cài đặt, mở cửa sổ dòng lệnh (đã được cài đặt môi trường Python) rồi thực thi mã lệnh:

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

<sup>6</sup>fastai Documentation, “Dynamic UNet”, <https://docs.fast.ai/vision.models.unet.html>

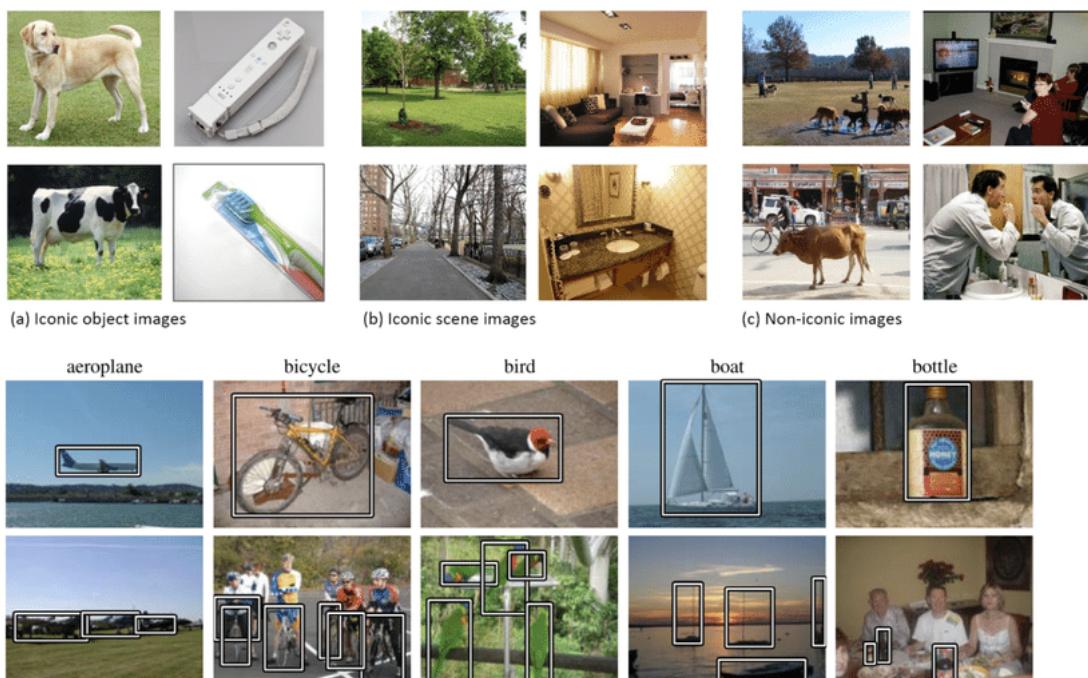
Toàn bộ mã nguồn, cũng như những tệp liên quan trong quá trình hoàn thành mô hình, có thể tìm thấy trong thư mục đồ án tại:

- Github: [https://github.com/dee-ex/EE3151\\_SEM202\\_PROJECT](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.

### 3.2 Tập dữ liệu

Tập dữ liệu được chọn sử dụng là tập **COCO** 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.



Hình 3.1: Hình ảnh trích từ tập dữ liệu COCO.

### 3.3 Chuẩn hoá và làm giàu dữ liệu

Chuẩn hoá dữ liệu đầu vào là bước tiền xử lí giúp tăng tốc độ hội huấn luyện cho mô hình, giảm sự phụ thuộc của gradient vào tỉ lệ các tham số và một số lợi ích khác [5, 2]. Ở đây, 3 kênh dữ liệu sẽ được chuẩn hoá cùng về một khoảng  $[-1, 1]$ :

$$\begin{aligned} \mathbf{L}^*_{\text{normalization}} &= \frac{\mathbf{L}^*}{50} - 1 \\ (\mathbf{a}^*, \mathbf{b}^*)_{\text{normalization}} &= \frac{(\mathbf{a}^*, \mathbf{b}^*)}{110} \end{aligned}$$

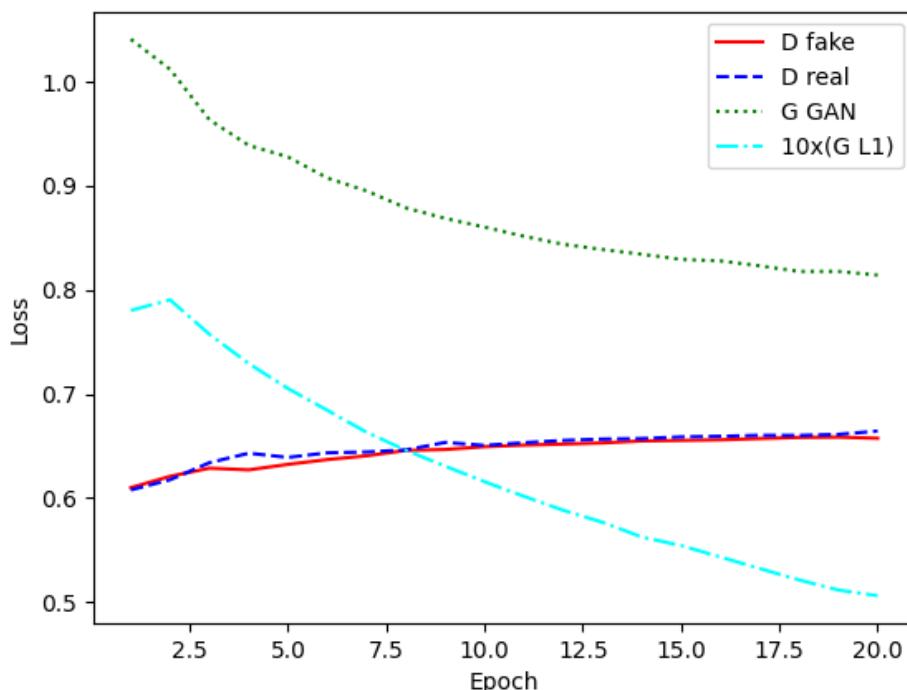
Bên cạnh đó, kích thước ảnh cũng sẽ được điều chỉnh lại thành ảnh vuông kích thước  $256 \times 256$  nếu cần thiết, và sử dụng lật đối xứng theo trực tung để làm giàu dữ liệu cho việc huấn luyện. Như vậy, tổng số lượng dữ liệu mà mô hình được học sẽ là  $10,000 \times 2 = 20,000$ .

### 3.4 Huấn luyện mô hình

Cách thức huấn luyện mô hình sử dụng Python-PyTorch được tham khảo chính từ [27, 31], đồng thời cũng có thêm một vài tham khảo về việc sử dụng Keras [26, 18, 4] để huấn luyện.

Vì mô hình bộ sinh phức tạp hơn rất nhiều so với bộ phân biệt, vậy nên để cho quá trình huấn luyện GAN được ổn định hơn, ta sẽ tiền huấn luyện [10] độc lập bộ sinh bằng  $\mathcal{L}_{\text{L1}}(G)$  qua 20 epoch với kích thước batch là 16. Mỗi epoch mất khoảng 6–7 phút khi huấn luyện trên Google Colab<sup>7</sup>. Thuật toán tối ưu được sử dụng là Adam [19] 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. Hệ số cân bằng được chọn  $\lambda = 100$ . Thuật toán tối ưu vẫn sẽ dùng là Adam với tốc độ học  $\alpha = 2 \cdot 10^{-4}$  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. Mỗi epoch mất khoảng 10–16 phút bằng việc sử dụng Google Colab để thực thi.



Hình 3.2: Giá trị các thành phần mất mát của bộ phân biệt và bộ sinh qua từng epoch.

Nhận thấy xu hướng của giá trị mất mát khi huấn luyện mô hình GAN là giá trị mất mát của bộ phân biệt tăng thì giá trị mất mát của bộ sinh sẽ giảm và cả hai dường như đều sẽ hội tụ về một giá trị. Lí giải điều này, 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. Điều này là hợp lý với nguyên lý non-zero-sum-games. Ngoài ra, giá trị mất mát của bộ sinh theo chuẩn 1 cũng được giảm, tuy không đáng kể.

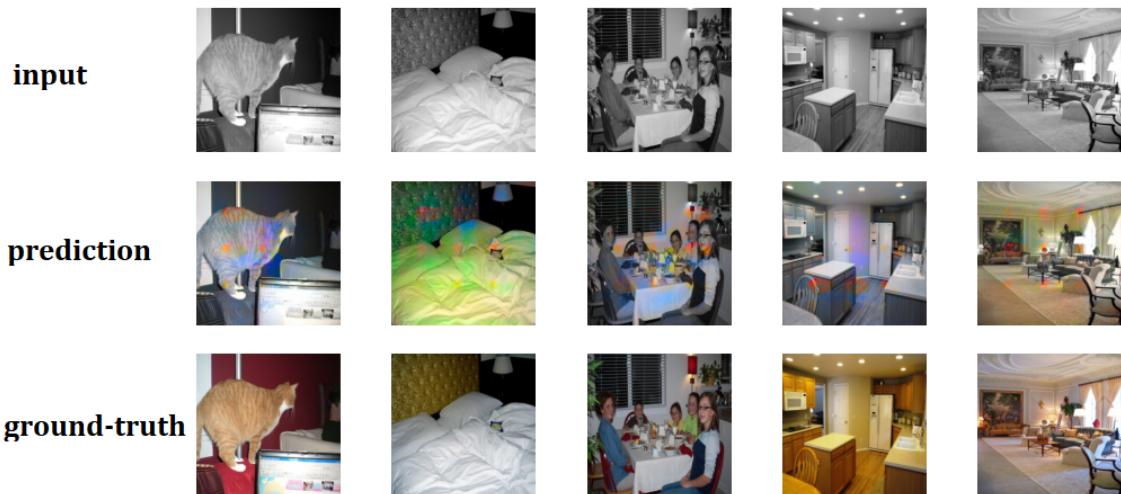
Sau một số thử nghiệm định tính đơn giản, mô hình có kết quả tốt nhất là mô hình sau epoch thứ 15. Và sẽ là lựa chọn cho các thử nghiệm trong những phần sau.

<sup>7</sup>Google Colab, <https://colab.research.google.com/>

## 4 Thủ nghiệm và đánh giá mô hình

### 4.1 Mô hình GAN với bộ sinh đơn giản

Kết quả ở hình dưới là kết quả của mô hình được thử nghiệm với 5 bức ảnh ngẫu nhiên lấy từ tập COCO (không nằm trong tập huấn luyện). Khá khôn may mắn, kết quả không được như chúng ta mong đợi.



Hình 4.1: 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 sau vài chục epoch. Nhưng, như đã nêu ra [2.6], 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.

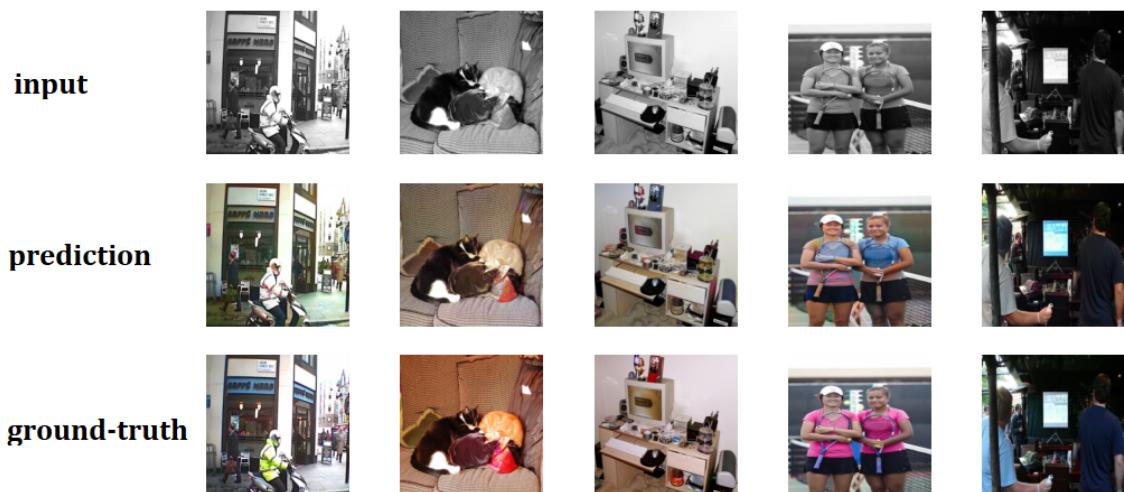
Nhìn thêm ở một góc độ khác, việc huấn luyện một mạng GAN cũng không hề dễ dàng, khi GAN rất nhạy cảm với những thông số được cài đặt khi huấn luyện. Sau nhiều lần thử nghiệm thất bại, một trong số những kinh nghiệm rút ra là, trong hầu hết các mô hình GAN, ta nên khởi tạo các trọng số của mô hình và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

### 4.2 Mô hình GAN với bộ sinh có xương sống ResNet18

Kết quả ở hình 4.2 dễ thấy đã cải thiện hơn hẳn so với kết quả ở hình 4.1. Màu tạo ra khi gắn với ảnh xám ít bị lem hay lộn xộn. Một vài đối tượng có màu khác với màu của ảnh gốc, ví dụ người lái xe ở ảnh phía trái ngoài cùng, nhưng vẫn hết sức hợp lý và tự nhiên.

Một vài 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. Để lý giải cho hạn chế này, có thể những vùng chưa có màu hợp lý là trong quá trình huấn luyện, mô hình chưa được học qua ảnh có đặc trưng, đối tượng tương tự.

Điều trên dẫn đến một hạn chế khác nữa là đối với những ảnh không có đặc trưng rõ ràng, những đặc trưng mà chưa được đưa vào huấn luyện cho mô hình. Chẳng hạn ảnh chủ đề logo hình 4.3, mô hình hoàn toàn không có khả năng tạo được màu phù hợp vì không hiểu được đặc trưng.



Hình 4.2: 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).



Hình 4.3: Mô hình hoàn toàn bối rối trước logo kỷ niệm 60 năm của trường DH Bách Khoa TP.HCM.



Hình 4.4: Trường hợp đặc biệt khi mô hình làm việc tệ với ảnh có đặc trưng đơn giản.

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 4.2, ảnh hai cô gái cầm vợt tennis (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 hình 4.4 thực sự là một kết quả bất ngờ, mà hiện tại vẫn chưa có một lí giải nào 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 [4.3] cũng có một 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.

### 4.3 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 mô hình GAN mang lại. Một số ảnh đen trắng 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 4.5: 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 đạt được khá đúng với những gì mong đợi.



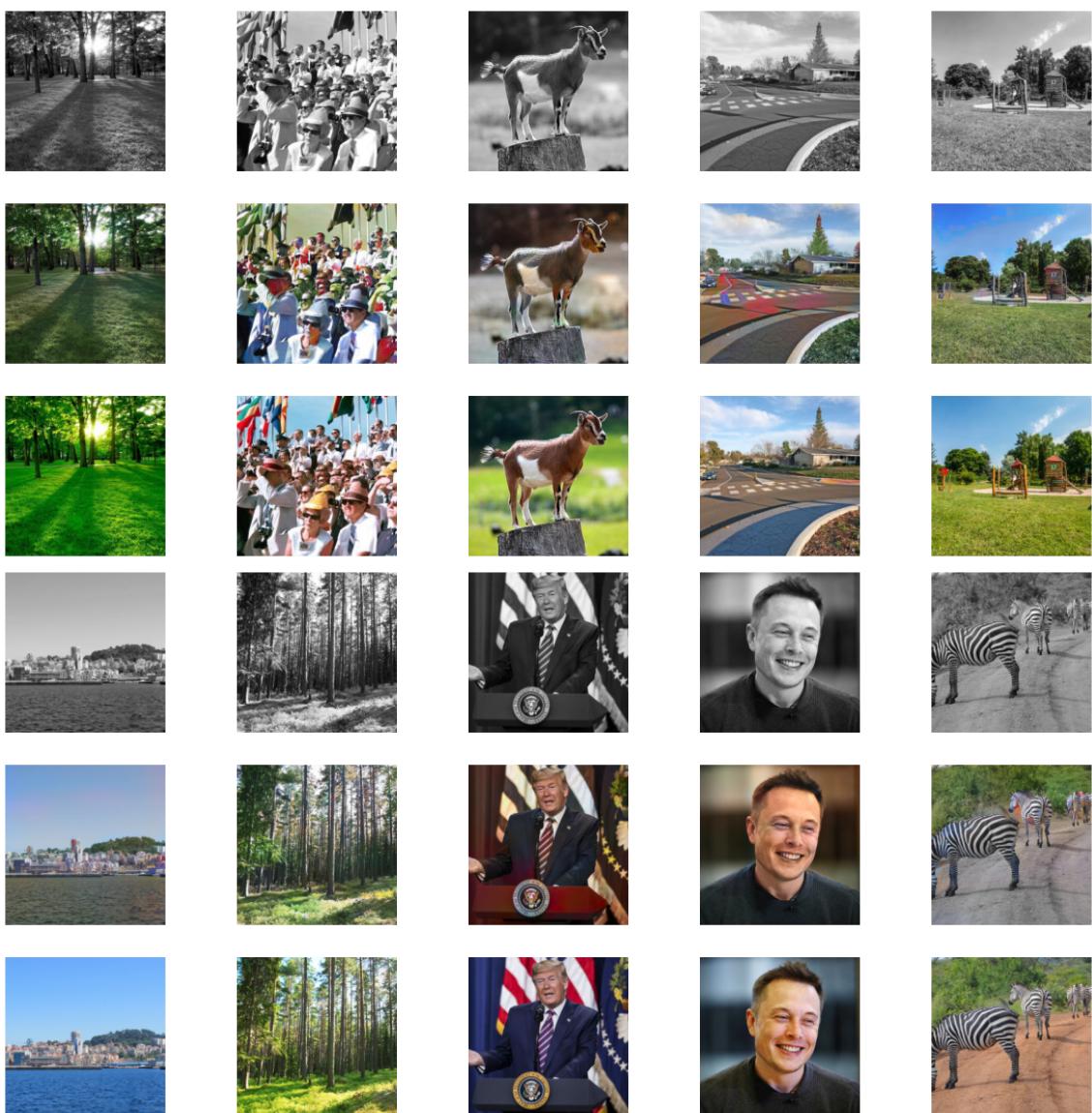
Hình 4.6: Phía trên và dưới lần lượt là kết quả của bộ sinh trước và sau khi huấn luyện đối nghịch.

Ta có thể thấy trong hình 4.6 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.

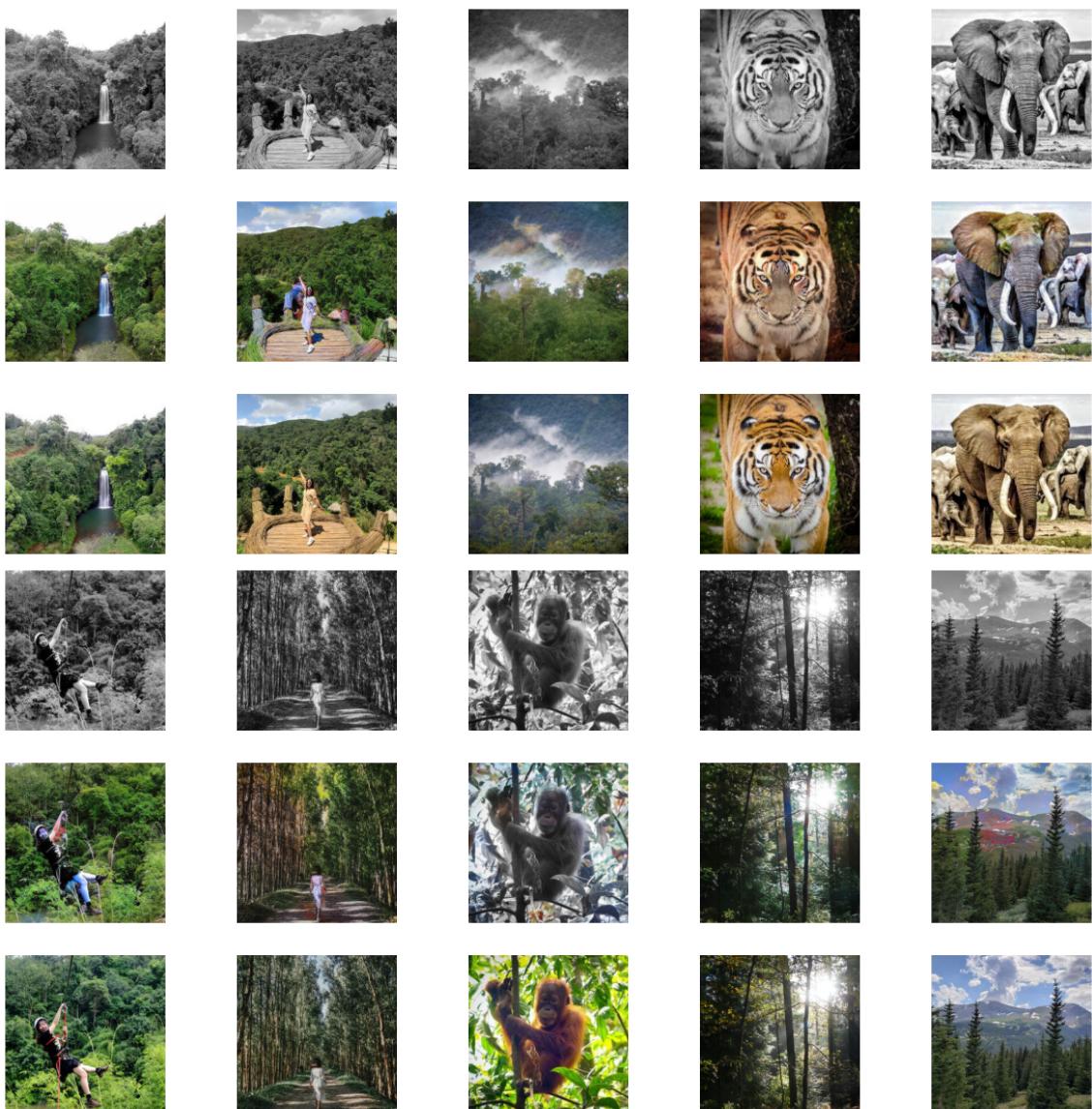
Điểm yếu của việc áp dụng đơn giản hàm măt măt so sánh sai lệch giữa các điểm ảnh, bằng cách dùng chuẩn bậc 1 hoặc bậc 2 như thông thường, đối với bài toán tạo màu rất khó để có kết quả tốt, vì mô hình lúc này được khuyến khích để dùng màu xám. Điều này có thể lí giải là mô hình học được một quy tắc, khi không chắc chắn nên chọn màu gì, thì chọn một giá trị trung bình ( $\mathbf{a}^*, \mathbf{b}^*$ )  $\approx (0, 0)$  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 sôh hơn màu xám dẫn, đến kết quả của mô hình trước khi đưa vào mạng GAN huấn luyện chưa đáp ứng mục tiêu hợp lí, chân thực đưa ra.

#### 4.4 Khảo sát đánh giá chất lượng mô hình



Hình 4.7: Một số kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc.

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



Hình 4.8: Thêm Một vài kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ánh 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 4.9: Phân mô tả của bài đánh giá.

giác con người. Một cuộc khảo sát đã được tiến hành bởi sự giúp đỡ của 100 người, bao gồm những sinh viên thuộc 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 4.7 và 4.8, 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.

The screenshot shows a Google Form interface. At the top, there is a question titled "Ảnh 01 \*". Below it is a photograph of a park path. A horizontal scale below the image has five points labeled 1, 2, 3, 4, and 5. Below each point is a radio button, with the first one (labeled 'rất giả') being selected. To the right of the scale is the label "rất thật". Below the question area, there is a summary section titled "100 responses". It shows a red bar indicating "Not accepting responses". A message for respondents states: "This form is no longer accepting responses". At the bottom of the summary section, there are three tabs: "Summary" (selected), "Question", and "Individual".

Hình 4.10: Phần câu hỏi và số lượng người tham gia cuộc khảo sát thực hiện thông qua Google Forms<sup>8</sup>.

Bảng 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 4.2.

Trong những ảnh được bình chọn, ảnh có số điểm cao nhất là ảnh thứ 05 với 4.01 (mức 4/5) theo sau đó là những ảnh có số điểm 3.96 (07) và 3.94 (09) (xem hình 4.11). Còn ảnh có số điểm thấp nhất là ảnh số 02 với 2.53 (mức 2-3/5), ngay phía trên là những ảnh có số điểm 2.75 (15) và 2.76 (20) (xem hình 4.12). 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 thứ 15. Chỉ

<sup>8</sup>Google Forms, <https://www.google.com/forms/about/>

Ảnh	Thang điểm	1	2	3	4	5	Tổng
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
<b>Tổng</b>		146	366	513	535	440	2000

Bảng 4.1: Kết quả bình chọn cho các ảnh.

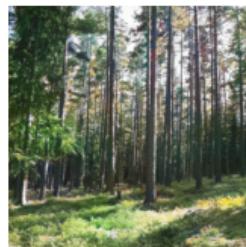
<b>Điểm trung bình</b>	Ảnh 01	Ảnh 02	Ảnh 03	Ảnh 04	Ảnh 05
	3.65	<b>2.53</b>	3.87	2.88	<b>4.01</b>
<b>Điểm trung bình</b>	Ảnh 06	Ảnh 07	Ảnh 08	Ảnh 09	Ảnh 10
	2.97	3.96	3.86	3.94	3.19
<b>Điểm trung bình</b>	Ảnh 11	Ảnh 12	Ảnh 13	Ảnh 14	Ảnh 15
	3.89	3.56	3.40	3.52	2.75
<b>Điểm trung bình</b>	Ảnh 16	Ảnh 17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

Bảng 4.2: Điểm trung bình của các ảnh.

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 kỳ tệ như ảnh 16 với số điểm 2.91 và ảnh số 02 với số điểm thấp nhất trong 20 bức ảnh khảo sát.



**4.01**



**3.96**



**3.94**

Hình 4.11: Những ảnh có điểm trung bình cao nhất.



**2.53**



**2.75**

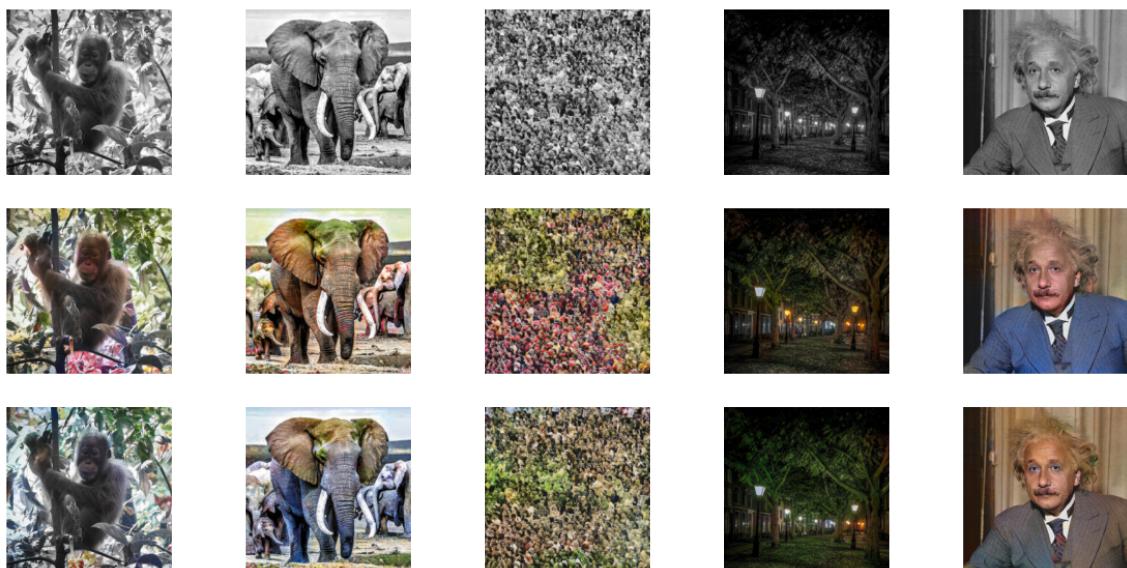


**2.76**

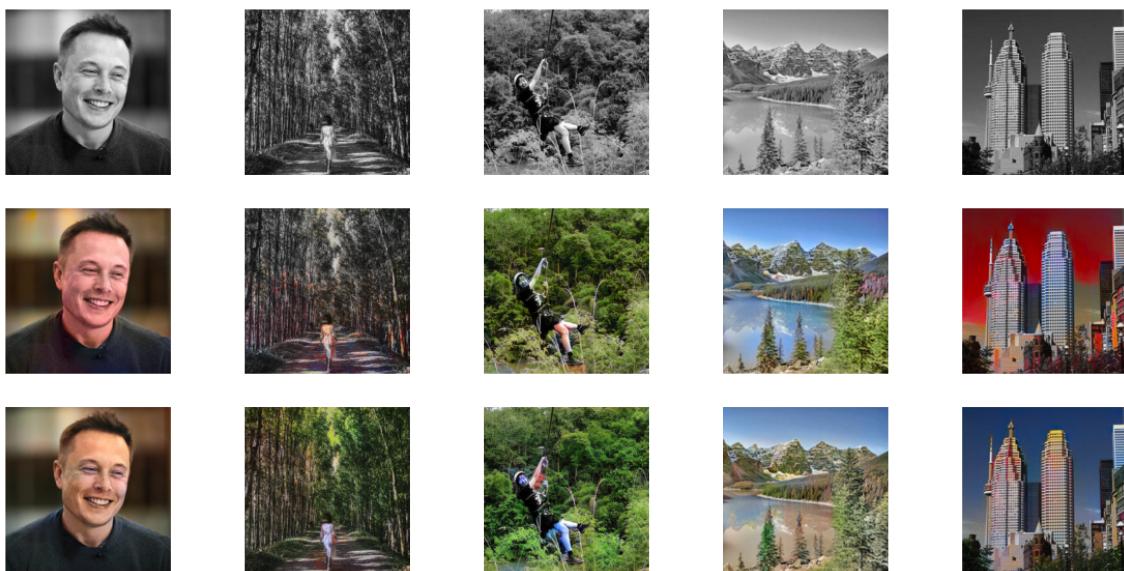
Hình 4.12: Những ảnh có điểm trung bình thấp nhất.

Tuy số lượng mẫu khảo sát nhỏ với chỉ 20 bức ảnh (chưa đủ làm tổng thể) và sự đa dạng không 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 thì mô hình có thể xem ở mức **trung bình-khá**.

#### 4.5 Kết quả sau khi huấn luyện lại với tập dữ liệu lớn hơn



Hình 4.13: So sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại.



Hình 4.14: 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.

Đây là mô hình được khởi tạo với kiến trúc và huấn luyện với các thông số hoàn toàn giống với mô hình mà ta đã thử nghiệm. Nhưng lại 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 4.13, cột 2; hình 4.14, 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 khá tươi sáng, khi có nhiều điểm ảnh được cho màu đỏ hoặc lam (màu áo ở hình 4.13, 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 4.13, cột thứ 4; màu nước ở hình 4.14, cột thứ 4). Nhưng cũng đầy rủi ro, khi lại dễ dàng gây ra sự lem luốc (màu trời, màu tòa nhà ở hình 4.14, cột thứ 5). Để cân đối được sáng tạo và khuôn khổ cho mô hình, công việc này cần thêm nhiều thử nghiệm trong tương lai.

## 5 Ứng dụng

### 5.1 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 <https://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. Vì mục đích thử nghiệm là chủ yếu, các phương thức bảo mật CSRF, CORS (xem hình 5.1) được tắt để tránh gây phiền hà khi thao tác.



Hình 5.1: Bảo mật CORS cản trở khi thử nghiệm API

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.

```
<anonymous>          app.c30df806293a47afa4b3.hot-update.js:1
  ↪ Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at http://localhost:8000/main/. (Reason: CORS header 'Access-Control-Allow-Origin' missing). [Learn More]
  ↪ Uncaught (in promise) TypeError: NetworkError when attempting to fetch resource.

>>
```

```
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>: Headers.prototype { 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(), ... }
```

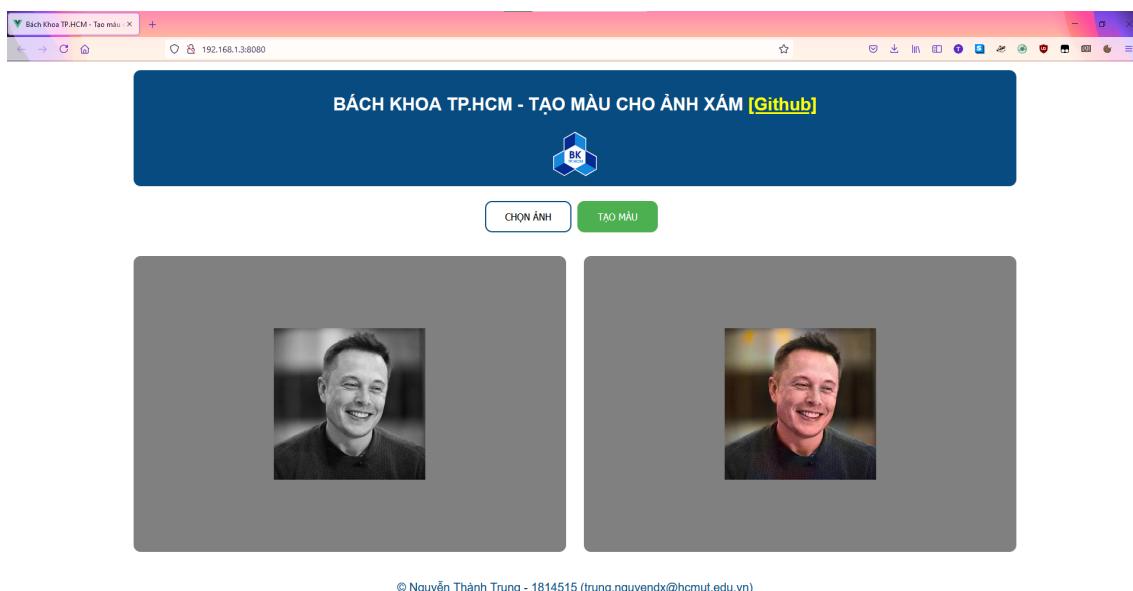
Hình 5.2: Phản hồi trả về của endpoint

The screenshot shows the Postman interface with a POST request to 'localhost:8000/main/'. The request body is set to 'form-data' with a key 'image' and value 'g1.png'. The response tab shows a preview of a smiling man's face, indicating the API successfully returned an image.

Hình 5.3: Thử nghiệm API trên phần mềm Postman

## 5.2 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. 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.



Hình 5.4: Ứng dụng web tạo màu đơn giản bằng VueJS kết hợp API

## 6 Tổng kết

### 6.1 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 dùng bộ phân biệt Patch  $70 \times 70$ , được huấn luyện theo kiểu đối nghịch kết hợp với một hàm mất mát chuẩn bậc 1 bằng tập dữ liệu 10,000 bức ảnh của COCO. Ánh xạ  $\mathcal{G} : \mathbf{L}^* \rightarrow (\mathbf{a}^*, \mathbf{b}^*)$  để tạo 2 kênh màu trong không gian màu  $\mathbf{L}^* \mathbf{a}^* \mathbf{b}^*$  chính là bộ sinh của mạng được thiết kế theo kiến trúc Unet có xương sống được học chuyển giao từ mô hình ResNet18 đã được tinh chỉnh.

Chất lượng mô hình 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. Theo như mục tiêu được đề ra [1.2], mô hình đã hoàn thành được ở mức trung bình-khá.

Bên cạnh đó, xây dựng được 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.

### 6.2 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, 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 [4.5]. Ngoài ra, sử dụng một xương sống được tinh chỉnh 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 hóa 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 [24], 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 [6] 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 [33] nhận thấy rằng ta có tận dụng phân bố phân lớp trong quá trình huấn luyện cho mạng GAN, và đưa ra mô hình với cái tên là ChromaGAN.

### 6.3 Hướng phát triển

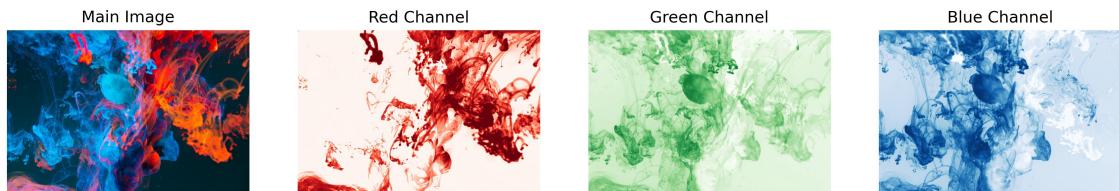
Đề tài có thể được áp dụng để làm những tác vụ liên quan đến phục chế màu cho ảnh xám, bị mất màu hoặc ảnh cũ xưa, thời chiến tranh. Giúp cho quá trình phục chế được tiết kiệm thời gian. Phối hợp với một ít tinh chỉnh bằng phương pháp thủ công, ta hoàn toàn có thể có những kết quả phục chế đẹp đẽ và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

Không chỉ dừng lại ở hình ảnh, 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 qua việc phục chế màu một đoạn nhỏ về phóng sự về chiến tranh ở Việt Nam.

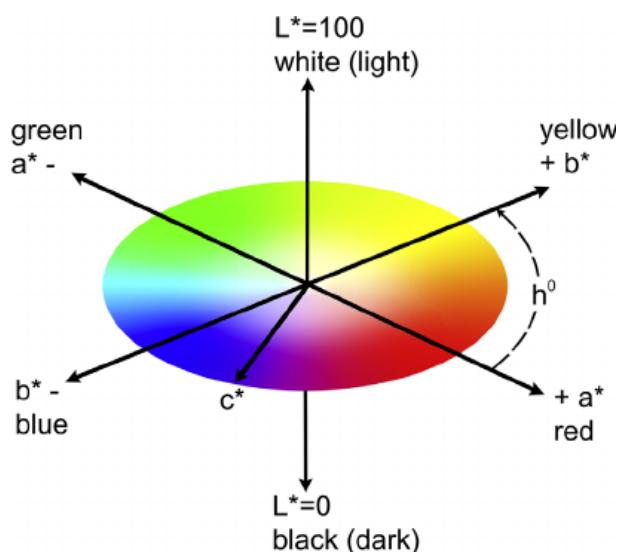


Hình 6.1: Ảnh Trường ĐH Bách Khoa TP.HCM ngày xưa khi chưa có màu (trên) và sau khi tô màu: mô hình ban đầu (trái), mô hình sau khi cải thiện (phải)

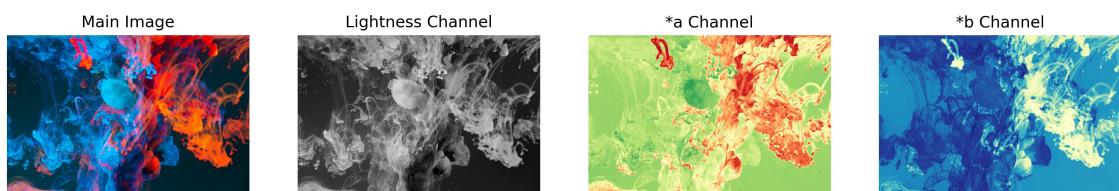
## 7 Phục lục - Ảnh màu



Hình 2.1: Các kênh màu đỏ, lục và lam được tách ra riêng biệt [31].



Hình 2.2: Không gian màu Lab.



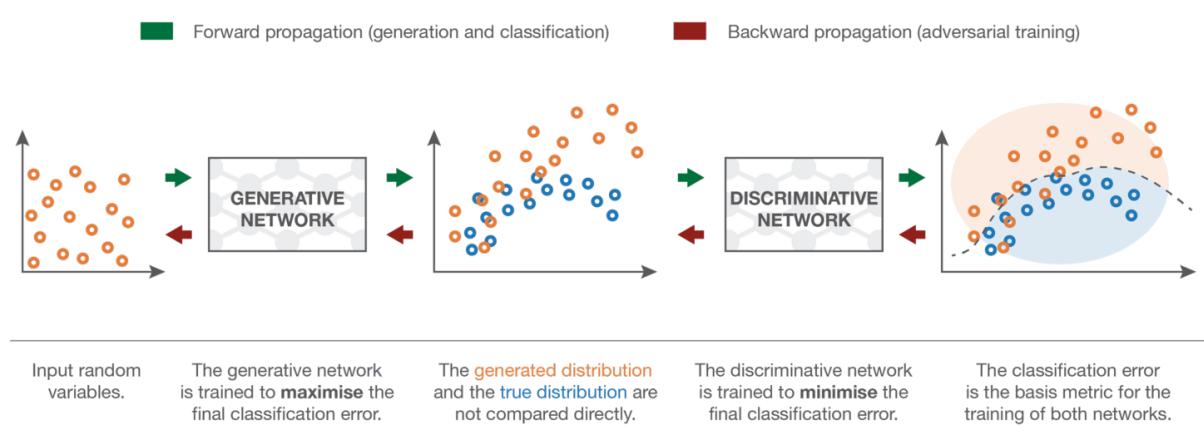
Hình 2.3: Các giá trị L, a và b được tách ra riêng biệt [31].



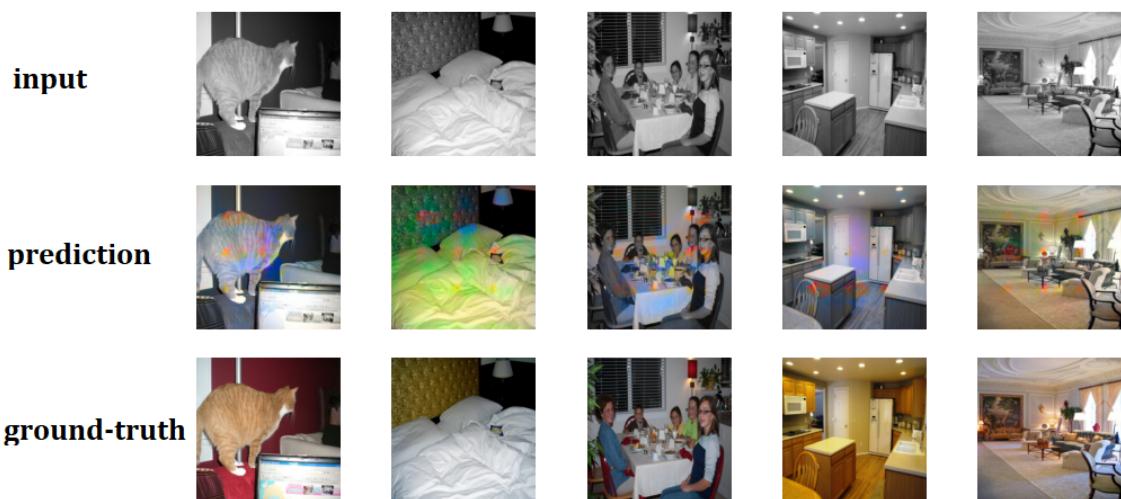
Hình 2.4: Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh [25].



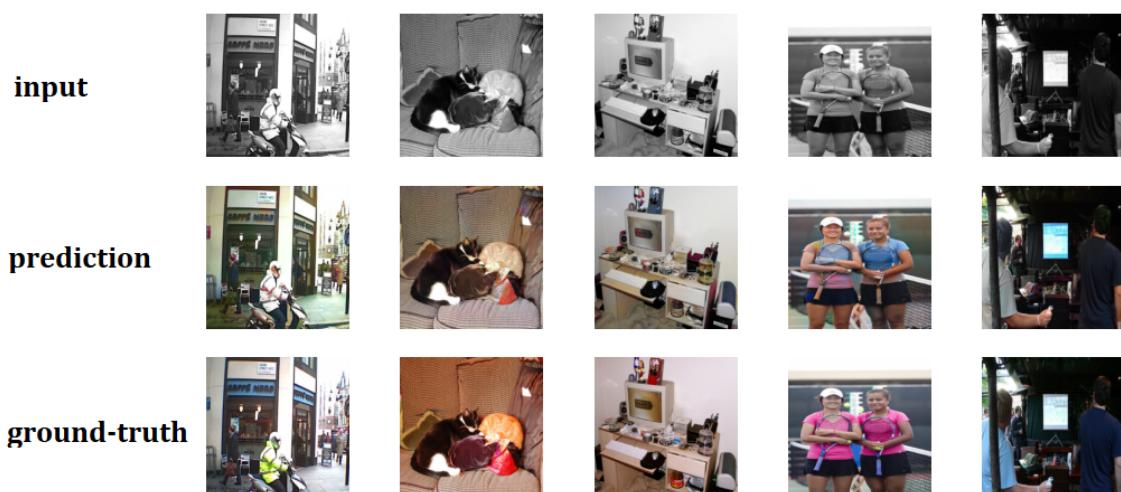
Hình 2.5: Mô tả ý tưởng thuật toán ảnh có bô cục tương tự [25].



Hình 2.9: Mô phỏng quá trình huấn luyện mạng GAN [29].



Hình 4.1: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản sau epoch thứ 34 trên tập dữ liệu COCO.



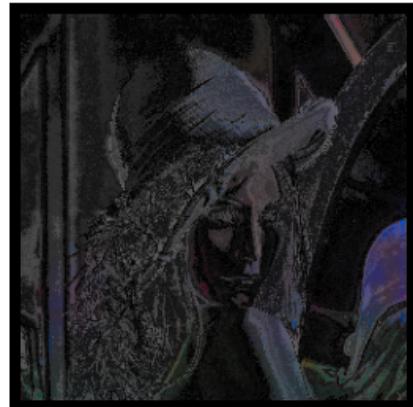
Hình 4.2: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản có xương sống ResNet18 sau epoch thứ 15 trên tập dữ liệu COCO (không nằm trong tập huấn luyện).



Hình 4.3: Mô hình hoàn toàn bế tắc trước logo kỷ niệm 60 năm của trường DH Bách Khoa TP.HCM.



input

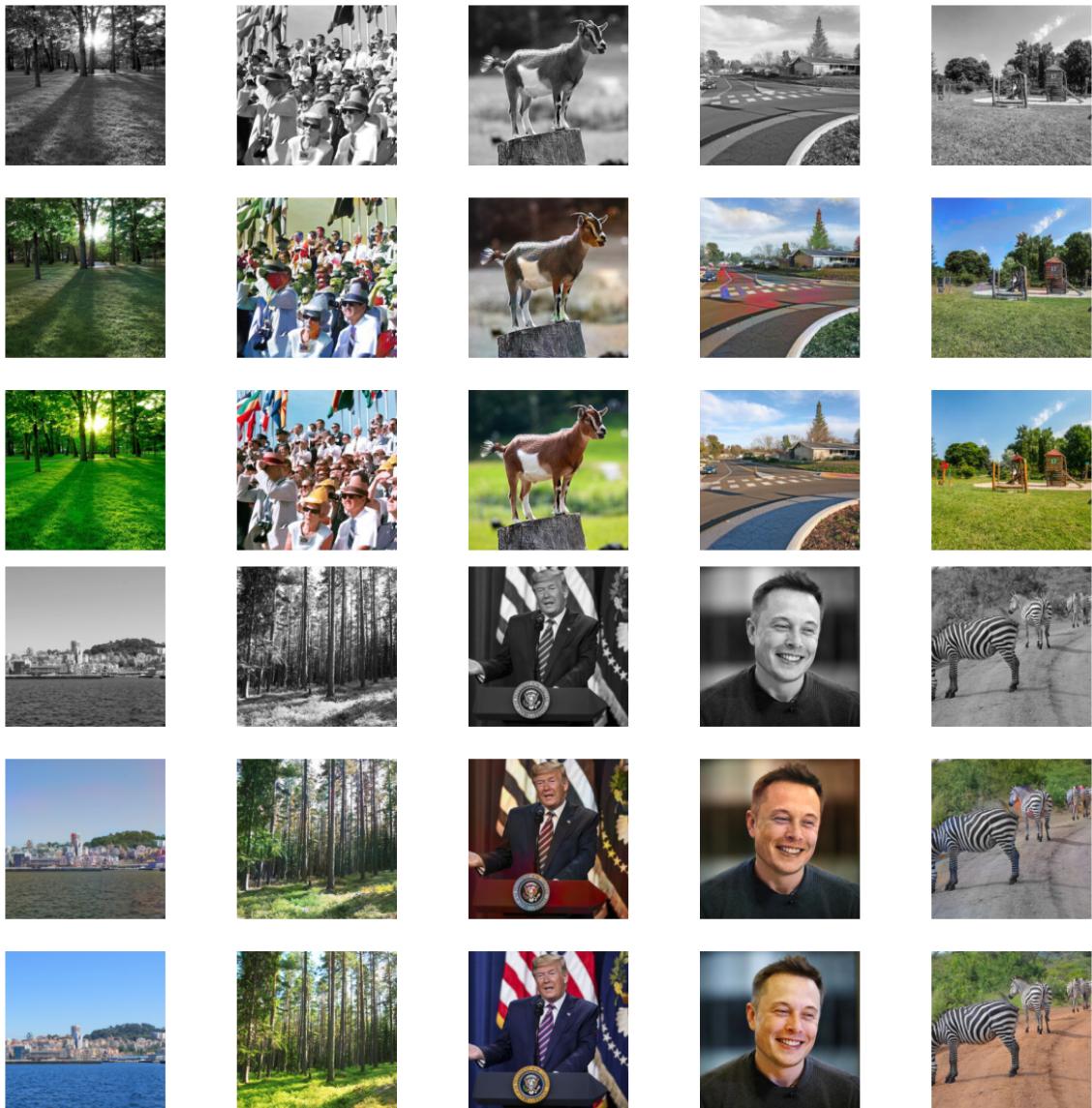


prediction

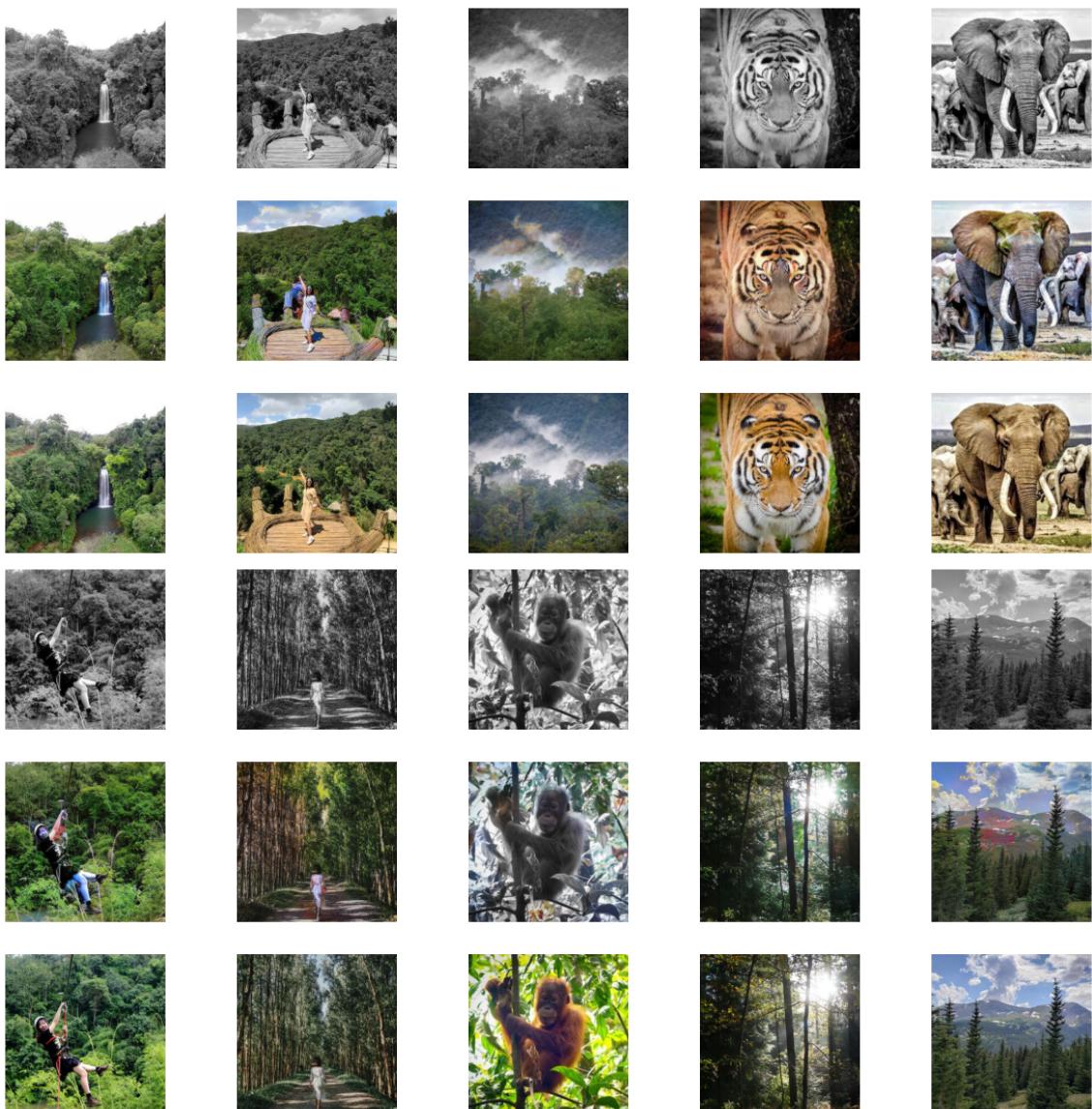
Hình 4.4: Trường hợp đặc biệt khi mô hình làm việc tệ với ảnh có đặc trưng đơn giản.



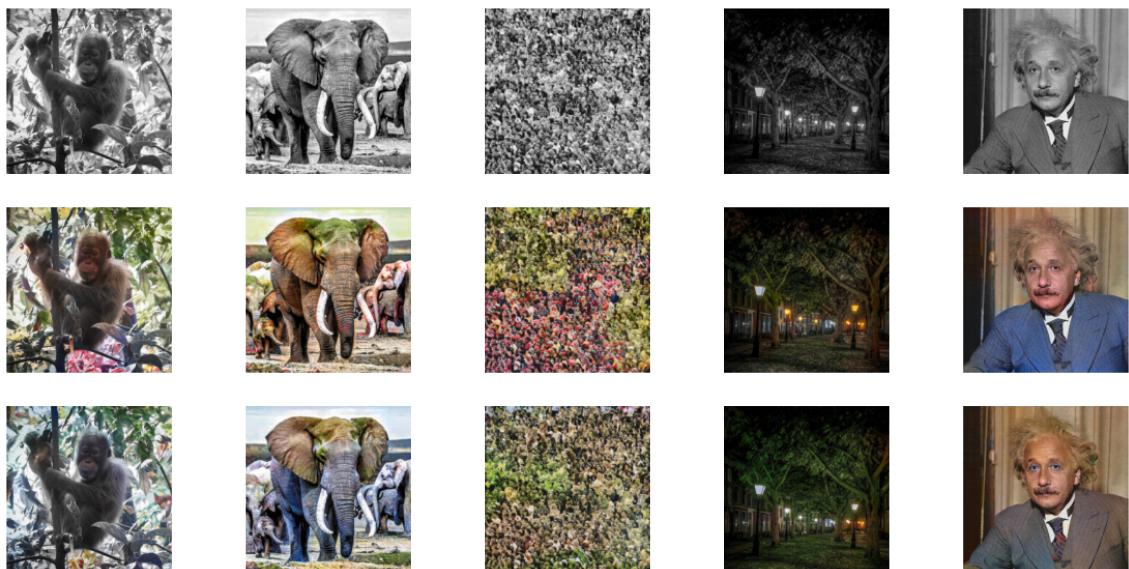
Hình 4.6: Phía trên và dưới lần lượt là kết quả của bộ sinh trước và sau khi huấn luyện đối nghịch.



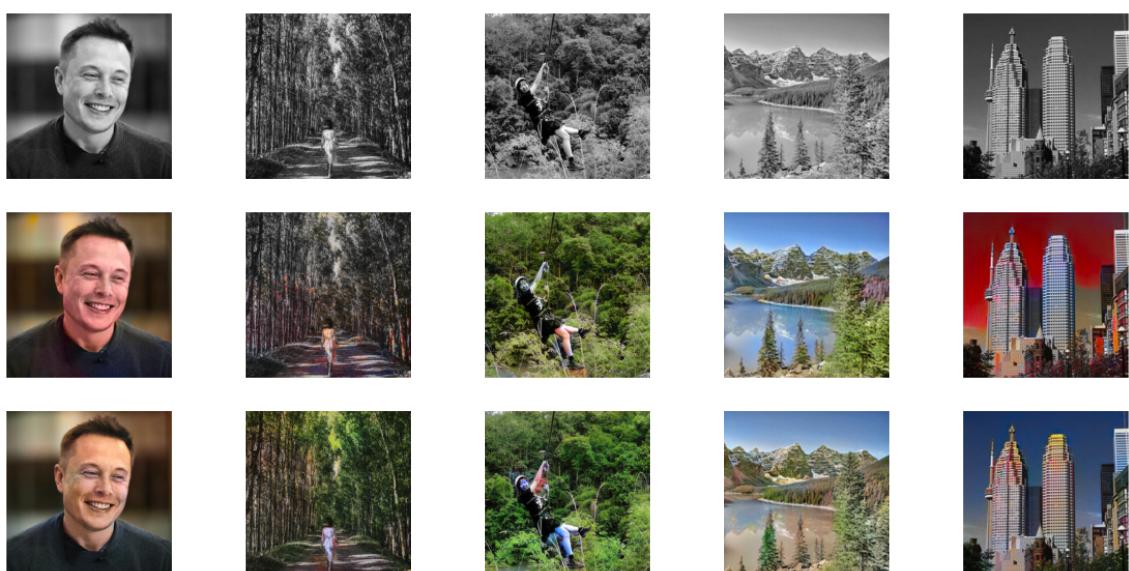
Hình 4.7: Một số kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ảnh sau khi tạo màu, ảnh gốc.



Hình 4.8: Thêm Một vài kết quả thử nghiệm khác. Các hàng lần lượt là ảnh xám, ánh sau khi tạo màu, ảnh gốc.



Hình 4.13 So sánh thử nghiệm sau khi huấn luyện lại. Hàng thứ 2 là kết quả của mô hình được huấn luyện lại.



Hình 4.14 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.



Hình 6.1 Ảnh Trường DH 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] The BrownBatman. "Reply: Calculate the Output size in Convolution layer". In: <https://stackoverflow.com/> (2018). URL: <https://bit.ly/3fn0Jrn>.
- [4] Jason Brownlee. "How to Develop a Pix2Pix GAN for Image-to-Image Translation". In: <https://machinelearningmastery.com/> (2019). URL: <https://bit.ly/3hVPOrY>.
- [5] Jason Brownlee. "How to use Data Scaling Improve Deep Learning Model Stability and Performance". In: <https://machinelearningmastery.com/> (2019). URL: <https://bit.ly/2QQQ1QK>.
- [6] Yun Cao et al. *Unsupervised Diverse Colorization via Generative Adversarial Networks*. 2017. arXiv: [1702.06674 \[cs.CV\]](https://arxiv.org/abs/1702.06674).
- [7] Z. Cheng, Q. Yang, and B. Sheng. "Deep colorization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 415–423.
- [8] Richard Gall. "Working Principles of Generative Adversarial Networks (GANs)". In: <https://dzone.com/> (2018). URL: <https://bit.ly/3ue1M3r>.
- [9] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: [1406.2661 \[stat.ML\]](https://arxiv.org/abs/1406.2661).
- [10] 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).
- [11] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: [1512.03385 \[cs.CV\]](https://arxiv.org/abs/1512.03385).
- [12] Gao Huang et al. *Densely Connected Convolutional Networks*. 2018. arXiv: [1608.06993 \[cs.CV\]](https://arxiv.org/abs/1608.06993).
- [13] 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.
- [14] R. Ironi, D. Cohen-Or, and D. Lischinski. "Colorization by example". In: *Rendering Techniques*. Citeseer. 2005, pp. 201–210.
- [15] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: [1611.07004 \[cs.CV\]](https://arxiv.org/abs/1611.07004).
- [16] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. arXiv: [1812.04948 \[cs.NE\]](https://arxiv.org/abs/1812.04948).
- [17] 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](https://doi.org/10.1007/s10462-020-09825-6). URL: <http://dx.doi.org/10.1007/s10462-020-09825-6>.
- [18] Pham Dinh Khanh. "Pix2Pix GAN". In: <https://phamdinhhkhanh.github.io/> (2020). URL: <https://bit.ly/3umnrRo>.

- [19] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: [1412.6980 \[cs.LG\]](https://arxiv.org/abs/1412.6980).
- [20] A. Levin, D. Lischinski, and Y. Weiss. “Colorization using optimization”. In: *ACM transactions on graphics (TOG)*. Vol. 23. ACM. 2004, pp. 689–694.
- [21] Q. Luan et al. “Natural image colorization”. In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association. 2007, pp. 309–320.
- [22] 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/2RIHhhe>.
- [23] Christopher Thomas BSc Hons. MIAP. “U-Nets with ResNet Encoders and cross connections”. In: <https://towardsdatascience.com/> (2019). URL: <https://bit.ly/3ttEu8x>.
- [24] 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](http://dx.doi.org/10.1007/978-3-319-94544-6_9).
- [25] Trung Thanh Nguyen. “Auto Colorize GrayScale Image With Deep Learning”. In: <https://forum.machinelearningcovan.com/> (2018). URL: <https://bit.ly/3eKdmwN>.
- [26] Tuan Nguyen. “Pix2pix”. In: [nttuan8.com/](https://nttuan8.com/) (2020). URL: <https://bit.ly/3xGVR9h>.
- [27] Aladdin Perssonl. *Pix2Pix implementation from scratch*. YouTube. 2021. URL: <https://youtu.be/oLvmLJkmXuc>.
- [28] Phillip. “Reply: Random noise z augmentation with cGAN”. In: <https://github.com/> (2017). URL: <https://bit.ly/3oM7EPE>.
- [29] Joseph Rocca. “Understanding Generative Adversarial Networks (GANs)”. In: <https://towardsdatascience.com/> (2019). URL: <https://bit.ly/3vzdPIU>.
- [30] 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).
- [31] Moein Shariatnia. “Colorizing black & white images with U-Net and conditional GAN - A Tutorial”. In: <https://towardsdatascience.com/> (2020). URL: <https://bit.ly/3eKaaBj>.
- [32] 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.
- [33] 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).
- [34] 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.
- [35] Wikipedia. “CIELAB color space”. In: <https://en.wikipedia.org/wiki/> (2021). URL: [https://en.wikipedia.org/wiki/CIELAB%5C\\_color%5C\\_space](https://en.wikipedia.org/wiki/CIELAB%5C_color%5C_space).
- [36] R. Zhang, P. Isola, and A. A. Efros. “Colorful image colorization”. In: *European conference on computer vision*. Springer. 2016, pp. 649–666.
- [37] 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/3hVP0rY>.