

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



TRÍ TUỆ NHÂN TẠO

---

Đồ án

# Phục hồi màu cho ảnh xám

---

GVHD: PGS.TS Hà Hoàng Kha (hhkha@hcmut.edu.vn)  
Sinh viên: Nguyễn Thành Trung - 1814515

Hồ Chí Minh, 05/2021



## Lời cảm ơn

Dầu tiên, em xin bày tỏ lòng biết ơn đến chân thành sâu sắc tới giáo viên hướng dẫn của em, thầy **PGS.TS Hà Hoàng Kha** - “giảng viên bộ môn Viễn Thông” đã trực tiếp giúp đỡ, hướng dẫn em hoàn thành đồ án này.

Dẫu vậy, do kiến thức chuyên ngành còn hạn chế nên trong quá trình nghiên cứu đề tài em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong được sự quan tâm, góp ý của các thầy/cô giảng viên bộ môn để đề tài của em được đầy đủ và hoàn chỉnh hơn.

Em xin chân thành cảm ơn.

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

Sinh Viên



## Tóm tắt đồ án

Đồ án với mục tiêu nghiên cứu, tìm ra một mô hình có khả năng tự động việc tô màu cho những ảnh xám, cũ đã mất màu để giúp cho quá trình phục chế màu được tiết kiệm thời gian. Đồ án được thực hiện nhờ vào việc áp dụng những kiến thức về **trí tuệ nhân tạo** (*artificial intelligence*), cụ thể là **học sâu** (*deep learning*).

Mô hình có khả năng **tự động tô màu những ảnh xám, cũ một cách chân thực, hợp lý**. Kiến trúc mô hình là kiến trúc của một **mạng đối nghịch tạo sinh** (*GAN - Generative Adversarial Network*) với **bộ phân biệt**  $70 \times 70$  kết hợp với **bộ sinh theo kiến trúc mạng Unet** được tạo nên từ phương pháp **học chuyển tiếp** từ **mạng ResNet18**.



## Mục lục

<b>1 Giới thiệu</b>	<b>1</b>
1.1 Tổng quát . . . . .	1
1.2 Mục tiêu . . . . .	1
<b>2 Xử lí bài toán</b>	<b>2</b>
2.1 Lựa chọn không gian màu . . . . .	2
2.1.1 Không gian màu RGB . . . . .	2
2.1.2 Không gian màu Lab . . . . .	2
2.1.3 Vì sao lựa chọn Lab thay vì RGB? . . . . .	3
2.2 Một số phương pháp đơn giản . . . . .	3
2.2.1 Dánh dấu màu vài điểm ảnh và dùng thuật toán lan . . . . .	3
2.2.2 Dựa vào màu của ảnh có bố cục tương tự . . . . .	3
2.2.3 Hạn chế . . . . .	3
2.3 Phương pháp sử dụng Mạng đối nghịch tạo sinh (GAN - Generative Adversarial Network) kết hợp mô hình mạng Unet . . . . .	4
2.3.1 Mạng đối nghịch tạo sinh - GAN . . . . .	4
2.3.2 Mô hình mạng Unet . . . . .	6
2.4 Kiến trúc mô hình chính . . . . .	7
<b>3 Triển khai mô hình</b>	<b>8</b>
3.1 Ngôn ngữ lập trình & Thư viện chính . . . . .	8
3.2 Tập dữ liệu . . . . .	8
3.3 Chuẩn hoá dữ liệu . . . . .	9
3.4 Xây dựng mô hình . . . . .	9
3.4.1 Bộ sinh . . . . .	9
3.4.2 Bộ phân biệt $70 \times 70$ . . . . .	10
3.4.3 Học chuyển tiếp (Transfer learning) . . . . .	11
3.4.4 Tiền huấn luyện độc lập bộ sinh ResNet . . . . .	11
3.4.5 Huấn luyện mạng GAN với bộ sinh ResNet . . . . .	12
<b>4 Thử nghiệm mô hình</b>	<b>13</b>
4.1 Mô hình GAN với bộ sinh đơn giản kết hợp chuẩn 1 . . . . .	13
4.2 Mô hình GAN với bộ sinh ResNet kết hợp chuẩn 1 . . . . .	13
4.3 So sánh bộ sinh ResNet trước và khi được huấn luyện đối nghịch . . . . .	14
<b>5 Kết luận &amp; Hướng phát triển</b>	<b>15</b>
5.1 Kết luận . . . . .	15
5.2 Hướng phát triển . . . . .	16
<b>6 Phục lục - Ảnh màu</b>	<b>17</b>



## Danh sách hình minh họa

1	Hình 1.1: Nải chuối có thể màu lục hoặc màu vàng. Nguồn: <a href="https://bit.ly/3e7nqRs">https://bit.ly/3e7nqRs</a> . . . . .	1
2	Hình 2.1: Các kênh màu đỏ, lục và lam được tách ra riêng biệt. Nguồn: <a href="https://bit.ly/3eKaaBj">https://bit.ly/3eKaaBj</a> . . . . .	2
3	Hình 2.2: Không gian màu Lab. Nguồn: <a href="https://bit.ly/3nFIQIp">https://bit.ly/3nFIQIp</a> . . . . .	2
4	Hình 2.3: Các giá trị <b>L</b> , <b>a</b> và <b>b</b> được tách ra riêng biệt. Nguồn: <a href="https://bit.ly/3eKaaBj">https://bit.ly/3eKaaBj</a> . . . . .	2
5	Hình 2.4: Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh. Nguồn: <a href="https://bit.ly/3eKdmwN">https://bit.ly/3eKdmwN</a> . . . . .	3
6	Hình 2.5: Mô tả ý tưởng thuật toán ảnh có bố cục tương tự. Nguồn: <a href="https://bit.ly/3eKdmwN">https://bit.ly/3eKdmwN</a> . . . . .	3
7	Hình 2.6: Ảnh mặt người sinh ra bởi StyleGAN. Nguồn: <a href="https://github.com/NVlabs/stylegan">https://github.com/NVlabs/stylegan</a> . . . . .	4
8	Hình 2.7: Minh họa bộ sinh và bộ phân biệt trong mạng GAN. Nguồn: <a href="https://bit.ly/3ue1M3r">https://bit.ly/3ue1M3r</a> . . . . .	5
9	Hình 2.8: Kiến trúc mạng đối nghịch tạo sinh . . . . .	5
10	Hình 2.9: Mô phỏng quá trình huấn luyện mạng GAN. Nguồn: <a href="https://bit.ly/3vzdPIU">https://bit.ly/3vzdPIU</a> . . . . .	6
11	Hình 2.10: Kiến trúc mô hình mạng Unet. Nguồn: <a href="https://arxiv.org/abs/1505.04597">https://arxiv.org/abs/1505.04597</a> . . . . .	7
12	Hình 2.11: Mô hình GAN dự đoán ảnh dựa vào đường viền. Nguồn: <a href="https://arxiv.org/abs/1611.07004">https://arxiv.org/abs/1611.07004</a> . . . . .	7
13	Hình 3.1: Hình ảnh trích từ tập dữ liệu COCO . . . . .	9
14	Hình 3.2: Một trong 2 lựa chọn cho bộ sinh. Mô hình Encoder-Decoder trái, Unet phải . . . . .	10
15	Hình 3.3: Mô tả vùng nhận thức qua các tầng. Nguồn: <a href="https://bit.ly/3xGaeLa">https://bit.ly/3xGaeLa</a> . . . . .	10
16	Hình 3.4: So sánh các mô hình mạng tích chập hiện đại hiện nay . . . . .	11
17	Hình 3.5: Giá trị mất mát của bộ sinh trên hai tập huấn luyện & xác thực qua từng epoch . . . . .	12
18	Hình 3.6: Giá trị các thành phần mất mát của bộ phân biệt và bộ sinh qua từng epoch . . . . .	12
19	Hình 4.1: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản kết hợp chuẩn 1 trên dữ liệu xác thực của tập dữ liệu <b>COCO</b> sau epoch thứ 34 . . . . .	13
20	Hình 4.2: Kết quả dự đoán của mô hình GAN với bộ sinh ResNet kết hợp chuẩn 1 trên dữ liệu xác thực của tập dữ liệu <b>COCO</b> sau epoch thứ 15 . . . . .	13
21	Hình 4.3: Mô hình hoàn toàn bế tắc trước logo kỷ niệm 60 năm của trường DH Bách Khoa TP.HCM	14
22	Hình 4.4: Trường hợp đặc biệt khi mô hình làm việc tệ với ảnh có bộ cục đơn giản . . . . .	14
23	Hình 4.5: Những tấm ảnh được lựa chọn để thử nghiệm sự khác biệt . . . . .	15
24	Hình 4.6: So sánh kết quả dự đoán của hai mô hình là mô hình chỉ dùng chuẩn 1 (phía trên) với mô hình GAN kết hợp chuẩn 1 (phía dưới) . . . . .	15
25	Hình 5.1: Ảnh Trường ĐH Bách Khoa TP.HCM ngày xưa (trái) và sau khi tô màu (phải) . . . . .	16

## 1 Giới thiệu

### 1.1 Tổng quát

Nhu cầu phục chế ảnh màu từ ảnh xám không phải là hiếm. Thông thường là ảnh do tác động của thời gian, làm mất hoặc sai màu. Đôi khi lại là hình ảnh được chụp ở thời xưa, thời đại chưa có công nghệ chụp ảnh màu. Hiện nay, hầu hết mọi người đều sử dụng phương pháp thủ công bằng tay. Tuy có sự giúp đỡ của các công cụ hỗ trợ chỉnh sửa hình ảnh, để có thể có một tấm ảnh đẹp cũng đòi hỏi mất nhiều thời gian, có thể lên đến cả tháng đối với những bức ảnh có nhiều chi tiết.

Thấy được những điều trên, em đã thử tìm kiếm các bài báo về cách thức có thể tự động phục chế màu cho ảnh bằng **máy học/học sâu** (*Machine Learning/Deep Learning*).<sup>1</sup> Từ đó xây dựng một mô hình tự động hoá việc tô màu.

Hầu hết ý tưởng cho mô hình của em tham khảo từ bài báo **Image-to-Image Translation with Condition Adversarial Networks**, hay còn được biết với cái tên **pix2pix**, đưa ra giải pháp cho những vấn đề về image-to-image và một trong số đó có liên quan tới việc tô màu ảnh.

### 1.2 Mục tiêu

Việc khôi phục màu cho ảnh đúng đắn như những gì được chụp là một chuyện gần như bất khả thi vì không ai có thể biết được thực sự màu của những đối tượng cần được tô màu một cách chắc chắn. Hơn thế nữa, việc một đổi tượng có thể có nhiều màu phù hợp là chuyên rất đổi bình thường. Ví dụ đơn giản như một quả chuối có thể màu lục khi chưa chín nhưng lại cũng có thể là màu vàng khi chín.



Hình 1.1: Nải chuối có thể màu lục hoặc màu vàng. Nguồn: <https://bit.ly/3e7nqRs>

Nhưng không vì thế mà ta có thể tô bất cứ màu nào cho bất cứ vật thể nào, giả sử như việc tô màu da người là màu xanh biển hoặc mặt trời thì màu xanh lá, trông không được đúng đắn cho lắm.

Trên những cơ sở đó, mục tiêu của em khi lên kế hoạch tìm hiểu và thực hiện đề tài này là xây dựng được một mô hình có thể dự đoán màu của những đối tượng trong ảnh xám một cách hợp lý hơn là chính xác tuyệt đối. Chất lượng dự đoán của mô hình cũng không mong đợi có thể bằng được so với làm thủ công, nhưng cũng tiết kiệm thời gian hơn rất nhiều.

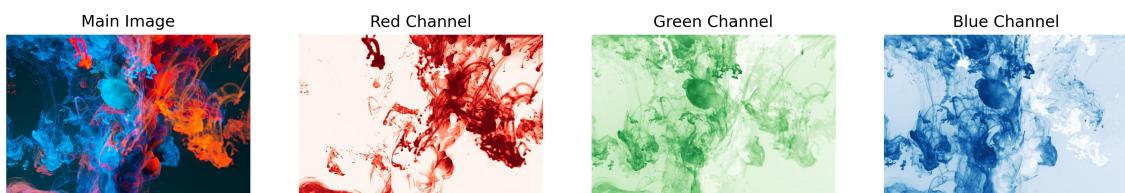
<sup>1</sup>Chi tiết trong phần **Tài liệu tham khảo**

## 2 Xử lý bài toán

### 2.1 Lựa chọn không gian màu

#### 2.1.1 Không gian màu RGB

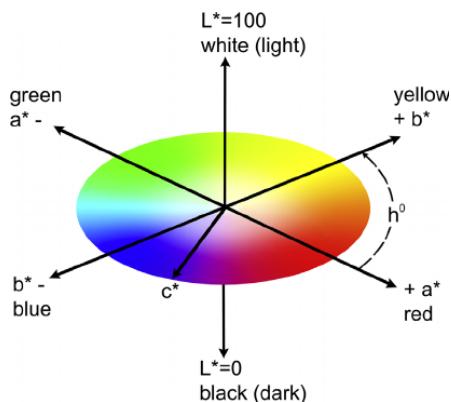
Hầu hết khi làm việc với ảnh số (*digital image*), ta sẽ thao tác với ảnh **RGB**, tức ảnh có 3 kênh màu đỏ-lục-lam (*red-green-blue*). Mỗi điểm ảnh (*pixel*) sẽ có 3 giá trị **R-G-B**, mỗi giá trị nằm trong đoạn [0, 255], tương ứng với 3 kênh màu để tạo nên được màu của chính điểm ảnh.



Hình 2.1: Các kênh màu đỏ, lục và lam được tách ra riêng biệt. Nguồn: <https://bit.ly/3eKaaBj>

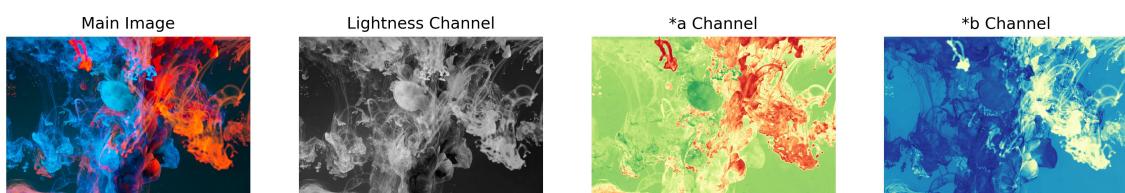
#### 2.1.2 Không gian màu Lab

Ngoài RGB, một không gian màu khác cũng được sử dụng nhiều là không gian màu **Lab**, không gian này cũng quan tâm đến 3 thông số của mỗi điểm ảnh. Thông số đầu tiên là **L** (*Lightness*) đại diện cho độ sáng của của mỗi điểm ảnh. 2 thông số còn lại là **a** và **b** sẽ mang thông tin lần lượt là lục-đỏ (*green-red*) và vàng-lam (*yellow-blue*). Giá trị **a** càng thấp thì lục nhiều, đỏ ít. Ngược lại thì lục ít, đỏ nhiều và tương tự với giá trị **b**.



Hình 2.2: Không gian màu Lab. Nguồn: <https://bit.ly/3nFIQIp>

Với giá trị **L**, giá trị sẽ nằm trong đoạn [0, 100]. Riêng 2 giá trị còn lại, không có cụ thể một khoảng nhất định, mà tùy thuộc vào phần mềm, chương trình mà ta sử dụng. Thường sẽ là đoạn [-128, 127], tuy nhiên mô hình của em không sử dụng khoảng giá trị trên. Chi tiết về vấn đề này sẽ được em làm rõ hơn ở phần **3.3 Chuẩn hoá dữ liệu**.



Hình 2.3: Các giá trị **L**, **a** và **b** được tách ra riêng biệt. Nguồn: <https://bit.ly/3eKaaBj>

### 2.1.3 Vì sao lựa chọn Lab thay vì RGB?

Sau một thời gian tìm hiểu qua các bài báo xử lý vẫn đề phục chế màu cho ảnh xám, em nhận thấy rằng hầu như không ai sử dụng không gian màu RGB để giải quyết bài toán này mà thay vào đó là sử dụng Lab. Lý do là khi sử dụng không gian màu Lab, sẽ có hai lợi thế:

- i) Không gian màu Lab được thiết kế để đồng nhất về cảm giác, điều này làm cho không gian màu này trở nên lý tưởng cho việc xử lý máy tính.
- ii) Ta có thể tận dụng kênh mức xám đầu vào **L** bằng cách điều chỉnh tỉ lệ (*scale*). Khi đó mô hình chỉ phải dự đoán kênh **a** và **b**.

## 2.2 Một số phương pháp đơn giản

### 2.2.1 Đánh dấu màu vài điểm ảnh và dùng thuật toán lan

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 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.



Hình 2.4: Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh. Nguồn: <https://bit.ly/3eKdmwN>

### 2.2.2 Dựa vào màu của ảnh có bối cảnh tương tự

Ý tưởng thuật toán khá giống các bài toán phân loại (*classification*), ta sẽ chọn ra một tấm ảnh có bối cảnh tương tự rồi dựa vào màu của ảnh đó kết hợp chỉnh sửa để đưa ra dự đoán.



Hình 2.5: Mô tả ý tưởng thuật toán ảnh có bối cảnh tương tự. Nguồn: <https://bit.ly/3eKdmwN>

Vì chưa thực sự tìm hiểu sâu về hai phương pháp trên, nên em xin phép không đi sâu vào phần cách thức thuật toán hoạt động.

### 2.2.3 Hạn chế

Cả hai phương pháp còn khá là thủ công, chưa tự động được khi ta phải vẽ màu định hướng hoặc là chọn ảnh có bối cảnh tương tự.

Với phương pháp **Đánh dấu màu vài điểm ảnh và dùng thuật toán lan** sẽ gặp khó khăn khi bức ảnh có nhiều chi tiết khác màu nhau thì sẽ gây nhiều khó khăn trong việc vẽ màu định hướng.

Việc tìm bối cảnh tương tự với bức ảnh đầu vào cũng là một bài toán không hề đơn giản, điều này khiến phương pháp **Dựa vào màu của ảnh có bối cảnh tương tự** trở nên bất khả thi khi chưa tìm được ảnh có bối cảnh tương tự ứng ý.

## 2.3 Phương pháp sử dụng Mạng đối nghịch tạo sinh (GAN - Generative Adversarial Network) kết hợp mô hình mạng Unet

Đây là phương pháp mà em đã chọn để xử lí cho bài toán này, để có thể hiểu được kiến trúc mô hình, ta sẽ xem xét những thành phần cơ bản của nó.

### 2.3.1 Mạng đối nghịch tạo sinh - 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 dưới đây là do GAN sinh ra, không phải mặt người thật.



Hình 2.6: Ảnh mặt người sinh ra bởi StyleGAN. Nguồn: <https://github.com/NVlabs/stylegan>

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

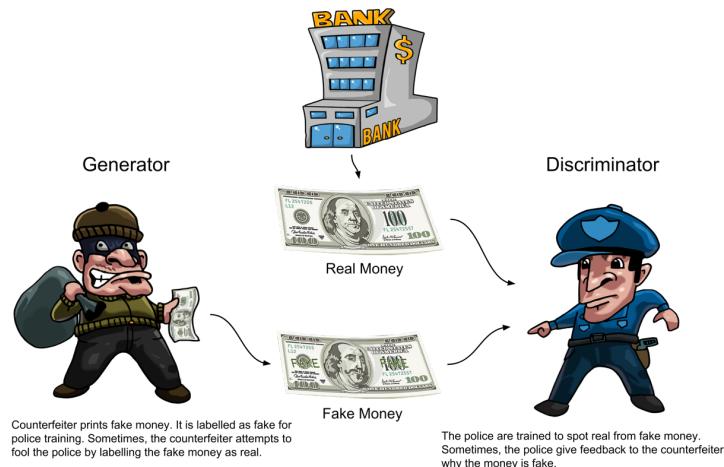
Trong quá trình huấn luyện mạng GAN thì nhiệm vụ của cảnh sát là học cách phân biệt tiền giả và tiền thật, bên cạnh đó là nói cho người làm tiền giả là nên làm giả như thế nào cho tốt hơn. Dần dần thì người làm tiền giả sẽ làm ra được tiền giống tiền thật và cảnh sát cũng trở nên thạo trong việc phân biệt tiền thật hay giả.

Ý tưởng của GAN bắt nguồn từ **Non-Zero-Sum Games**<sup>2</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 hoá cơ hội thắng của mình và tối thiểu hoá 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 (*Nash equilibrium*<sup>3</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

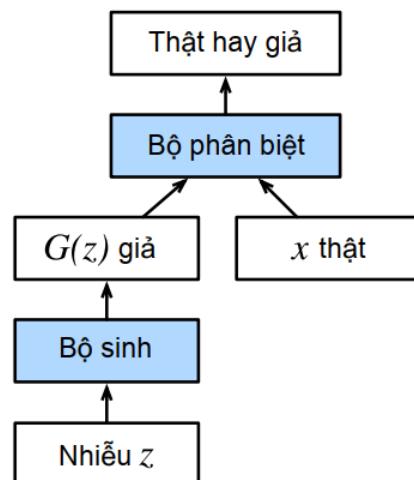
<sup>2</sup>Stanford, ‘Non-Zero-Sum Games’, <https://stanford.io/3nCiLKq>

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



Hình 2.7: Minh họa bộ sinh và bộ phân biệt trong mạng GAN. Nguồn: <https://bit.ly/3ue1M3r>

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>4</sup> để cung cấp tín hiệu cho việc huấn luyện cho bộ sinh. Điều này cho phép ta cải thiện bộ sinh có thể sinh dữ liệu tới khi ra được thứ gì đó giống như dữ liệu thực. Ở mức tối thiểu nhất, nó cần lựa 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 (*binary classification*) nhằm phân biệt đầu vào  $\mathbf{x}$  là thật (từ dữ liệu thật) hoặc giả (từ bộ sinh), được học theo kiểu giám sát (*supervised learning*). Thông thường, đầu ra của bộ phân biệt là một số vô hướng (*scalar*)  $o \in \mathbb{R}$  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, được học theo kiểu không giám sát (*unsupervised learning*), 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>5</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

<sup>4</sup>Wikipedia, ‘Two-sample hypothesis testing’, [https://en.wikipedia.org/wiki/Two-sample\\_hypothesis\\_testing](https://en.wikipedia.org/wiki/Two-sample_hypothesis_testing)

<sup>5</sup>Reddit, “Why is Z-dimension for GANs usually 100?”, <https://bit.ly/33nXNpk>

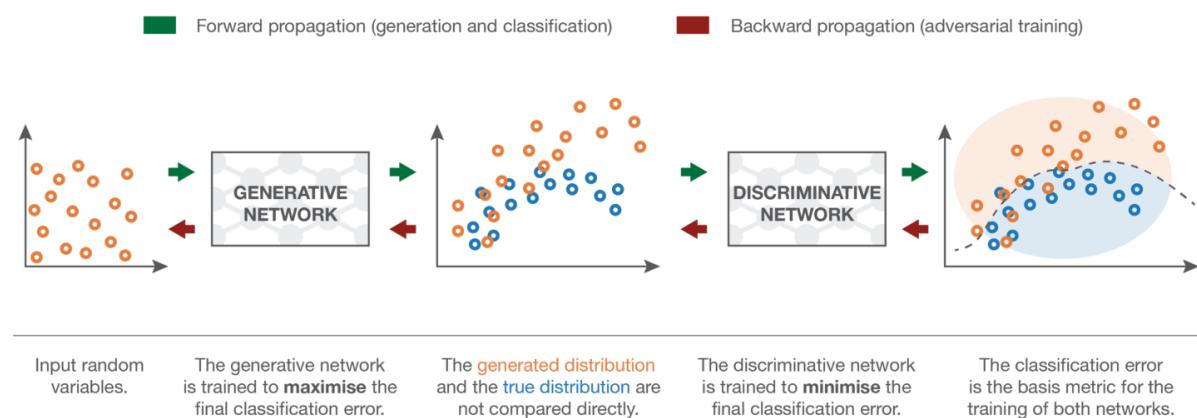
số của bộ sinh  $G$  nhằm cực đại hóa mất mát entropy chéo khi  $y = 0$ , tức là:

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

Tóm lại,  $D$  và  $G$  đang chơi trò “minimax” (cực tiểu hoá cực đại) với một hàm mục tiêu toàn diện 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, hiểu đơn giản là lấy trung bình của tất cả dữ liệu.



Hình 2.9: Mô phỏng quá trình huấn luyện mạng GAN. Nguồn: <https://bit.ly/3vzdPIU>

### 2.3.2 Mô hình mạng Unet

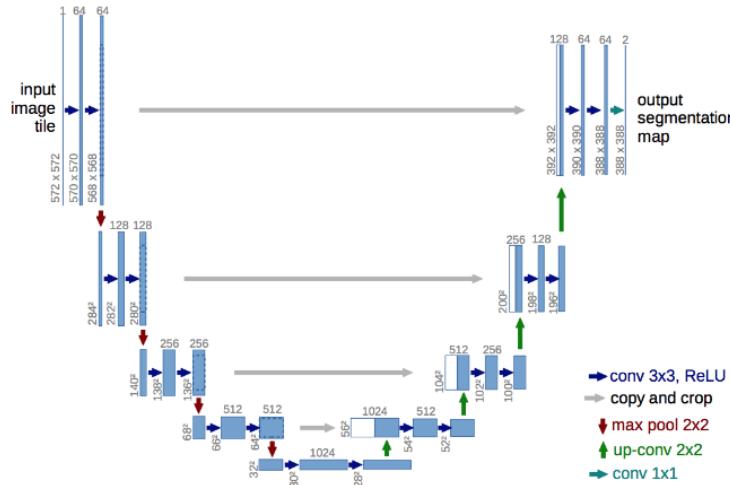
Unet là một kiến trúc được phát triển nhằm phân vùng các cấu trúc nơ-ron thần kinh trong não người. Kiến trúc này lần đầu được áp dụng đã giành chiến thắng trong cuộc thi EM segmentation challenge at ISBI 2012.<sup>6</sup> Mạng Unet bao gồm 2 nhánh đối xứng nhau hình chữ U nên được gọi là Unet. Kiến trúc mạng bao gồm 2 phần là **thu hẹp** (*contraction/encoder*) ở nhánh trái và **phân mở rộng** (*expansion/decoder*) 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 encoder. 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, hiểu nôm na là giảm độ phân giải (*resolution*) bằng cách sử dụng **gộp cực đại** (*max pooling*). Quá trình làm giảm kích thước còn được gọi là *downsampling*.
- Phần mở rộng: Gồm các tầng đối xứng tương ứng với các tầng của nhánh thu hẹp. Quá trình làm tăng kích thước tầng gọi là *upsampling*, hiểu theo một cách đơn giản là hàm ngược của tích chập (*Deconvolution*) để tăng lại độ phân giải của ảnh để biết được vị trí (*WHERE*) từ đó đánh dấu nhãn của từng điểm ảnh. 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** (*dilation convolution*). Nhưng được sử dụng nhiều nhất là **tích chập chuyển vị** (*transposed convolution*)

Đặc trưng riêng trong cấu trúc của Unet đó là áp dụng những **kết nối tắt** (*skip connection*) đối xứng giữa các tầng của nhánh bên trái tương ứng với các tầng bên nhánh bên phải.

Mô hình mạng Unet này sẽ được áp dụng vào bộ sinh trong mô hình mạng GAN. Hay nói cách khác, ta sẽ dùng mạng Unet để làm bộ sinh.

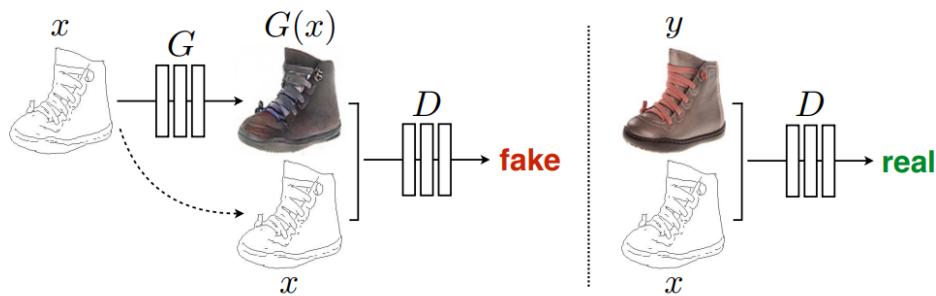
<sup>6</sup>ISBS Challenge: Segmentation of neuronal structures in EM stacks, [http://brainiac2.mit.edu/isbi\\_challenge/home](http://brainiac2.mit.edu/isbi_challenge/home)



Hình 2.10: Kiến trúc mô hình mạng Unet. Nguồn: <https://arxiv.org/abs/1505.04597>

## 2.4 Kiến trúc mô hình chính

Với bài toán tô màu thì mục tiêu là từ đầu vào  $\mathbf{L}$  (độ sáng - mức xám) của từng điểm ảnh ta sẽ dự đoán được cặp (**a**, **b**) để kết hợp tạo nên một tấm trong không gian màu Lab. Công việc này sẽ được bộ sinh đảm nhận. Như vậy, để bộ phân biệt phát hiện được đâu là thật giả, ta cần phải đưa vào cho bộ phân biệt cả đầu vào lẫn đầu ra tức là cặp dữ liệu - nhãn. Đây chính là điều kiện xác suất cho dự đoán của bộ phân biệt và cũng chính là điểm khác biệt so với GAN thông thường, khi bộ phân biệt được nhìn thấy dữ liệu đầu vào thay vì chỉ mỗi bộ sinh.



Hình 2.11: Mô hình GAN dự đoán ảnh dựa vào đường viền. Nguồn: <https://arxiv.org/abs/1611.07004>

Do đó, hàm mất mát của mô hình GAN chúng ta cũng sẽ khác biệt một chút so với một mạng GAN thông thường, cụ thể là:

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

Trong đó,  $\mathbf{x} \in \mathbb{R}^{1 \times S \times S}$  là đầu vào chính là mức xám  $\mathbf{L}$ , còn  $\mathbf{y} \in \mathbb{R}^{2 \times S \times S}$  đầu ra là (**a**, **b**). Nhiều  $\mathbf{z}$  được tạo ra từ việc **bỏ học** (*dropout*).<sup>7</sup>

Nếu ta không dùng kỹ thuật bỏ học, mô hình vẫn sẽ có thể được huấn luyện và đưa ra kết quả dự đoán tốt tuy nhiên không được tổng quát, tức kết quả dự đoán của mô hình sẽ khá bị bó buộc khi phân phối sẽ chỉ có thể là hàm delta.<sup>8</sup>

Một điểm khác biệt 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 là bộ phân biệt cục bộ (*Patch Discriminator*). Lí do là thay vì chỉ

<sup>7</sup>Cụ thể được đề cập trong bài báo “Image-to-Image Translation with Conditional Adversarial Networks” là “Instead, for our final models, we provide noise only in the form of dropout, applied on several layers of our generator at both training and test time.”

<sup>8</sup>Cụ thể được đề cập trong bài báo “Image-to-Image Translation with Conditional Adversarial Networks” là “Without  $z$ , the net could still learn a mapping from  $x$  to  $y$ , but would produce deterministic outputs, and therefore fail to match any distribution other than a delta function.”

trả về là một đầu ra là một số vô hướng thì bộ phân biệt cục bộ sẽ trả về một ma trận  $\mathbf{D} \in \mathbb{R}^{k \times k}$  bằng cách phân loại từng vùng  $N \times N$  của đầu vào (ảnh có màu) là thật hay giả hay vì toàn bộ một lúc. Những vùng  $N \times N$  này được gọi là vùng nhận thức (*receptive field*) và phân loại mỗi vùng nhận thức sẽ cho một giá trị  $\mathbf{D}_{ij}$ . Kích thước của ma trận  $\mathbf{D}$  (tức  $k$ ) cũng như là vùng nhận thức (tức  $N$ ) phụ thuộc vào các thông số của các tầng tích chập mà ta xây dựng cho bộ sinh, chi tiết sẽ được đề cập và làm rõ trong phần **3.4 Xây dựng mô hình**.

GAN hay chính xác là bộ sinh sẽ làm nhiệm vụ giúp cho màu chúng ta dự đoán được đẹp, giống như thật. Không chỉ dừng lại đó, để có thể kiểm soát được chuyên mô hình tô màu được đẹp và giống thật, ta sẽ cho bộ sinh kết hợp với hàm mất mát dùng chuẩn 1 (*norm 1 - L1 - mean absolute error*):

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

Tại sao lại là chuẩn 1 thay vì chuẩn 2 (chuẩn Euclid)? Lí do là chuẩn 2 có thường “trùng phạt” mất mát nhiều hơn so với chuẩn 1, do đó điều này làm bó buộc kết quả dự đoán của bộ sinh làm cho mô hình có xu thế bảo thủ, sẽ dự đoán bằng cách lấy giá trị trung bình. Thay vì vậy, chuẩn 1 lại làm mềm việc “trùng phạt” này lại, khiến cho mô hình có thể sáng tạo để đưa ra dự đoán “phiêu lưu” hơn.

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 khá lớn, và thường giá trị được chọn sẽ là  $\lambda = 100$ .

## 3 Triển khai mô hình

### 3.1 Ngôn ngữ lập trình & Thư viện chính

Em sử dụng ngôn ngữ **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/>

Bên cạnh đó, em cũng sử dụng thêm thư viện **fastai** (bản mới nhất) cho việc xây dựng mạng Unet. Để cài đặt, ta sử dụng lệnh sau trong môi trường Python:

```
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 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**<sup>9</sup> trong đường dẫn phía trên.

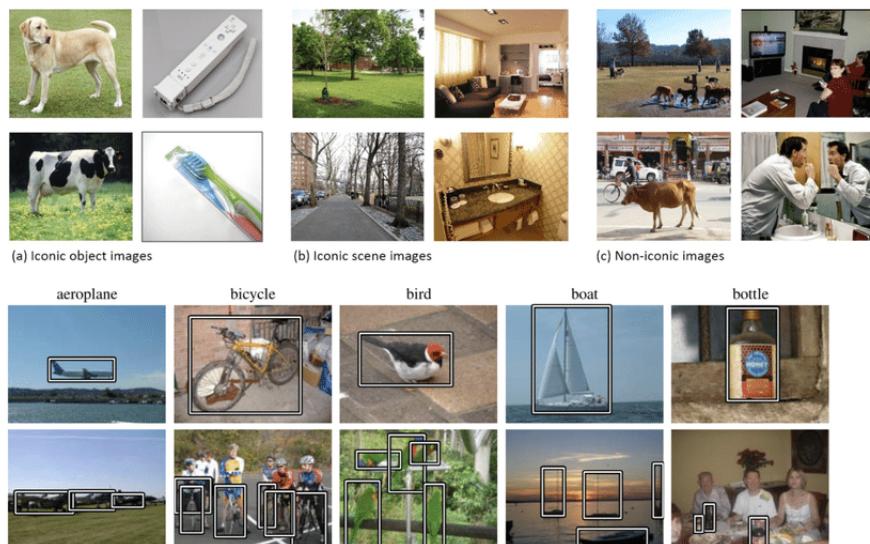
### 3.2 Tập dữ liệu

Tập dữ liệu được em chọn sử dụng là tập **COCO** có trong thư viện **fastai**, ta có thể dễ dàng tải về. Em quyết định chọn 10,000 ảnh để huấn luyện, còn quá trình xác thực sẽ là 2,000.

Mô hình được chia ra huấn luyện hai giai đoạn. Giai đoạn đầu tiên là tiền huấn luyện, sẽ chỉ huấn luyện riêng bộ sinh với hàm mất mát  $\mathcal{L}_{L1}$ . Lúc này, sẽ dùng 2,000 tấm ảnh cho việc xác thực để đưa ra mô hình phù hợp cho giai đoạn huấn luyện sau. Giai đoạn tiếp theo là huấn luyện GAN, vì yêu cầu mô hình sẽ là tô màu cho ảnh một cách hợp lý chứ không phải chính xác tuyệt đối, chỉ một vài tấm ảnh được lấy ra trong tập xác thực sẽ được đánh giá kết quả tô màu qua từng epoch để quyết định lựa chọn mô hình cuối cùng.

Riêng với phần thử nghiệm, dữ liệu sẽ được em chọn để có thể chủ động trong việc lựa chọn bối cảnh cho ảnh, từ đó có đánh giá đa dạng nhất cho mô hình.

<sup>9</sup> Được chứa trong thư mục [https://github.com/dee-ex/EE3151\\_SEM202\\_PROJECT](https://github.com/dee-ex/EE3151_SEM202_PROJECT)



Hình 3.1: Hình ảnh trích từ tập dữ liệu COCO

### 3.3 Chuẩn hoá dữ liệu

Khi sử dụng Pytorch để xử lý hình ảnh nhờ vào thư viện PIL và skimage, thì khoảng giá trị của không gian màu xám khi đưa về Tensor sẽ là  $[0, 1]$ , còn không gian màu Lab sẽ là:

- $\mathbf{L} \in [0, 100]$
- $\mathbf{a} \in [-86.1830, 98.2331]$
- $\mathbf{b} \in [-107.8573, 94.4781]$

Ta có thể kiểm tra chuyện đó bằng cách:

---

```
In [2]: import numpy as np
In [3]: colors = np.mgrid[0:256, 0:256, 0:256].astype(np.uint8)
In [4]: colors.shape
Out[4]: (3, 256, 256, 256)

In [6]: all_rgb = np.transpose(colors)
In [7]: from skimage import color
In [8]: all_lab = color.rgb2lab(all_rgb)
In [9]: np.max(all_lab, axis=(0, 1, 2))
Out[9]: array([100.        ,  98.23305386,  94.47812228])

In [10]: np.min(all_lab, axis=(0, 1, 2))
Out[10]: array([ 0.        , -86.18302974, -107.85730021])
```

---

Em quyết định sẽ chuẩn hoá cả đầu vào và đầu ra nằm trong khoảng  $[-1, 1]$  để áp dụng hàm kích hoạt đầu ra là hàm tanh;  $\sigma(z) = (e^z - e^{-z}) / (e^z + e^{-z}), R = (-1, 1)$ . Vậy nên, công thức chuẩn hoá sẽ là

$$\mathbf{L}_{\text{normalization}} = \frac{\mathbf{L}}{50} - 1$$

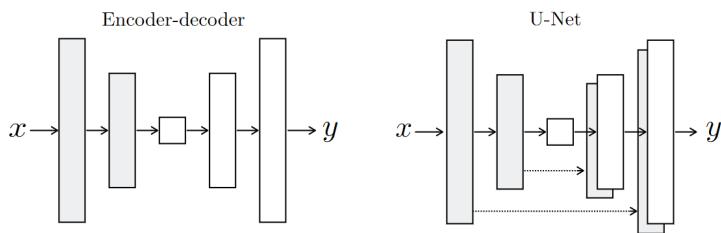
$$(\mathbf{a}, \mathbf{b})_{\text{normalization}} = \frac{(\mathbf{a}, \mathbf{b})}{110}$$

Bên cạnh đó, kích thước ảnh cũng sẽ được điều chỉnh (*resize*) lại thành ảnh vuông kích thước  $256 \times 256$ .

### 3.4 Xây dựng mô hình

#### 3.4.1 Bộ sinh

Tổng quát, bộ sinh sẽ nhận đầu vào là một tập gồm  $M \in \mathbb{N}^*$  ảnh xám kích thước  $(S, S) = 256 \times 256$ , tức là một Tensor  $\mathbf{X} \in \mathbb{R}^{M \times 1 \times S \times S}$ . Đầu ra của bộ sinh là một Tensor  $\mathbf{Y} \in \mathbb{R}^{M \times 2 \times S \times S}$ .



Hình 3.2: Một trong 2 lựa chọn cho bộ sinh. Mô hình Encoder-Decoding trái, Unet phải

Giả sử  $C_k$  là khối Convolution-BatchNorm-ReLU với  $k$  bộ lọc (kernel/filter).  $CD_k$  là khối Convolution-BatchNorm-Dropout-ReLU với tỉ lệ dropout là 50% và  $k$  bộ lọc. Tất cả đều sử dụng bộ lọc kích thước  $4 \times 4$  với sải bước (stride) là 2 và đệm (padding) là 1.

Sau mỗi khối  $C_k$  sẽ là một khối gộp cực đại để giảm kích thước (không tính chiều sâu) dữ liệu xuống gấp đôi. Còn sau mỗi khối  $CD_k$  sẽ là một khối giải chập thì kích thước (không tính chiều sâu) dữ liệu tăng gấp đôi. Cụ thể, mô hình sẽ có kiến trúc như chi tiết như sau:

- **encoder:** C64-C128-C256-C512-C512-C512-C512
- **decoder:** CD512-CD1024-CD1024-C1024-C512-C256-C128

Ở **encoder**, ta sẽ sử dụng hàm LeakyReLU với độ dốc âm (*negative slope*) là 0.2 thay cho ReLU. Tầng cuối cùng của bộ sinh sẽ là một hàm tanh. Một ngoại lệ là ở tầng đầu tiên ở **encoder** ta sẽ không sử dụng BatchNorm. Ta có lưu ý nhỏ ở đây, bản thân BatchNorm đã có bias nên là những tầng tích chập không cần phải có bias.<sup>10</sup>

### 3.4.2 Bộ phân biệt $70 \times 70$

Nhìn chung, bộ phân biệt thường không có kiến trúc quá phức tạp (nhiều tầng) và mô hình này cũng không phải là ngoại lệ. Bộ sinh sẽ gồm các khối:

- C64-C128-C256-C512

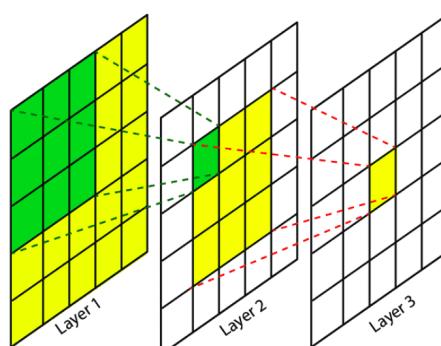
Thêm một lưu ý ở đây là ở khối tích chập C512 thì sải bước là 1. Sau đó, ta sẽ đưa kết quả đó qua thêm 1 tầng tích chập với chỉ một bộ lọc để đưa về một Tensor dạng  $(N, 1, 30, 30)$  và sải bước vẫn tiếp tục là 1.

Với kiến trúc như trên, ta sẽ được vùng nhận thức  $N \times N$  là  $70 \times 70$ . Đây là vùng nhận thức có kết quả thực nghiệm tốt nhất mà bài báo **pix2pix** đã chỉ ra.

Để tính kích thước vùng nhận thức ở tầng thứ  $l$  bất kì thì ta sẽ có công thức:<sup>11</sup>

$$K_{l-1} = K + s_l - 1(K_l - 1)$$

Trong đó  $K_l$  là kích thước vùng nhận thức ở tầng thứ  $l$ ,  $K$  là kích thước bộ lọc và  $s_l$  chính là số sải bước ở tầng thứ  $l$ .



Hình 3.3: Mô tả vùng nhận thức qua các tầng. Nguồn: <https://bit.ly/3xGaeLa>

<sup>10</sup>Stack Overflow - Patwie, “Can not use both bias and batch normalization in convolution layers”, <https://bit.ly/3f25eb2>

<sup>11</sup>Github - phillipi, “receptive\_field\_sizes.m”, [https://github.com/phillipi/pix2pix/blob/master/scripts/receptive\\_field\\_sizes.m](https://github.com/phillipi/pix2pix/blob/master/scripts/receptive_field_sizes.m)

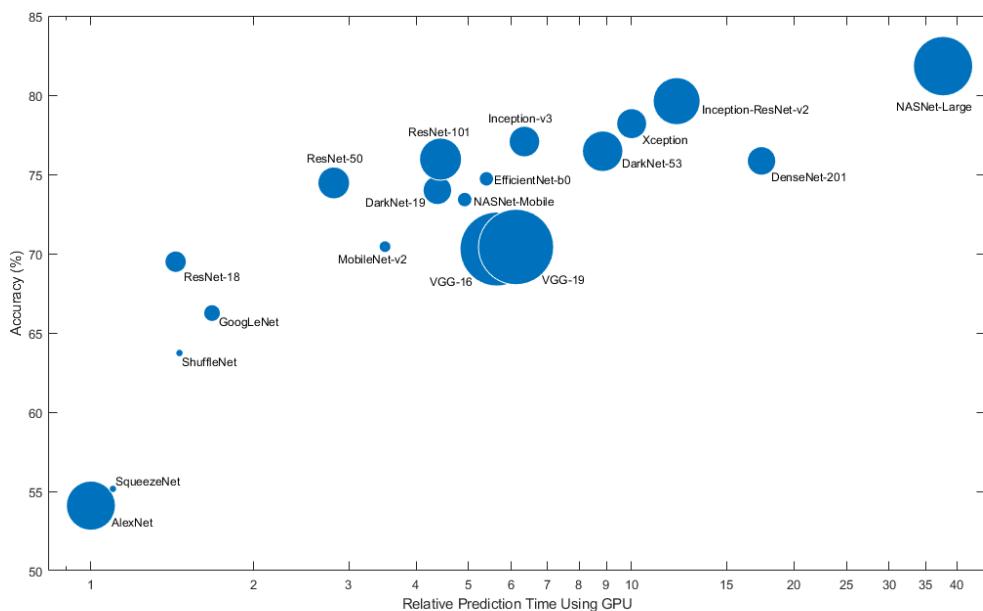
Như vậy, với mỗi đầu ra  $1 \times 1$  của kết quả  $30 \times 30$  thì kích thước vùng nhận thức quét được có thể được tính như sau:

$$\begin{aligned}K_4 &= K + s_4(K_5 - 1) = 4 + 1(1 - 1) = 4 \\K_3 &= K + s_3(K_4 - 1) = 4 + 1(4 - 1) = 7 \\K_2 &= K + s_2(K_3 - 1) = 4 + 2(7 - 1) = 16 \\K_1 &= K + s_1(K_2 - 1) = 4 + 2(16 - 1) = 34 \\K_0 &= K + s_0(K_1 - 1) = 4 + 2(34 - 1) = 70\end{aligned}$$

### 3.4.3 Học chuyển tiếp (Transfer learning)

Sau khi huấn luyện mô hình và thử nghiệm thì thấy kết quả không được khả quan cho lắm.<sup>12</sup> Do đó, em đã quyết định tìm kiếm một mạng Unet được huấn luyện sẵn. Một trong những giải pháp được áp dụng phổ biến đó là sử dụng mạng **Resnet** (**Residual Network**) cho phần thu hẹp (phía trái chữ U). Lý do là mạng ResNet cũng sử dụng những kết nối tắt trong kiến trúc của nó, phù hợp với kiến trúc mạng Unet. Ngoài ResNet, một mạng khác cũng có thể được áp dụng đó là **DenseNet**. Tuy nhiên, mạng **ResNet18** là lựa chọn cuối cùng do mô hình mạng có kích thước vừa phải, thuận tiện và thích hợp cho việc huấn luyện đối với phần cứng hiện tại của em.

Về phía phải chữ U, phần mở rộng, thư viện **fastai** cung cấp công cụ để có thể tự động xây dựng phần tương ứng với phần thu hẹp.



Hình 3.4: So sánh các mô hình mạng tích chập hiện đại hiện nay

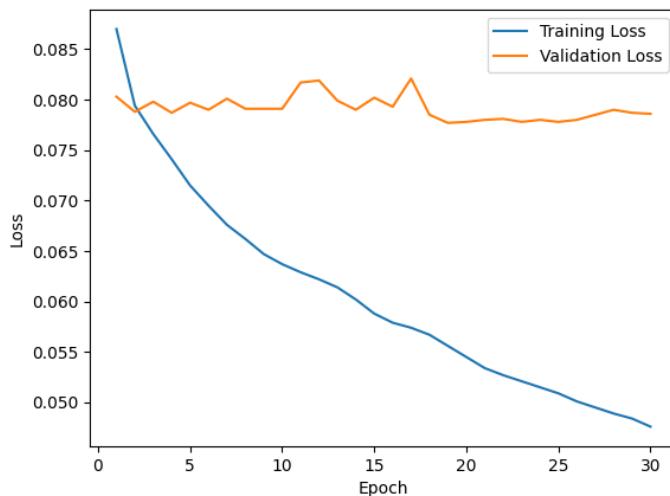
### 3.4.4 Tiên huấn luyện độc lập bộ sinh ResNet

Khi huấn luyện GAN, việc khó hơn cả là huấn luyện cho bộ sinh có thể sinh được dữ liệu ta mong muốn. Để tối ưu việc đó, ta sẽ đưa riêng ra huấn luyện bộ sinh trước khi đưa vào GAN nhờ vào hàm mất mát  $\mathcal{L}_{\text{L1}}(G)$ .

Việc huấn luyện, em sử dụng thuật toán tối ưu Adam với tốc độ học  $\alpha = 10^{-4}$ , tốc độ phân rã (*exponential decay rates*)  $\beta_1 = 0.9, \beta_2 = 0.999$  và  $\epsilon = 10^{-8}$ . Cho chạy batch size là 16 và 30 epoch. Mỗi epoch mất khoảng 6-7 phút (chưa tính phần xác thực).

Dựa vào **Hình 3.5**, giá trị mất mát của mô hình không cao. Bên cạnh đó, ta thấy rằng giá trị mất mát trên tập huấn luyện có xu hướng giảm và tiếp tục giảm, còn giá trị mất mát trên tập xác thực thì có vẻ đã bão hòa. Để tránh **quá khớp** (*overfitting*), em sẽ chọn mô hình thứ 20 (tức sau khi huấn luyện qua 20 epoch).

<sup>12</sup>Sẽ được trình bày trong phần 4 **Thử nghiệm mô hình**



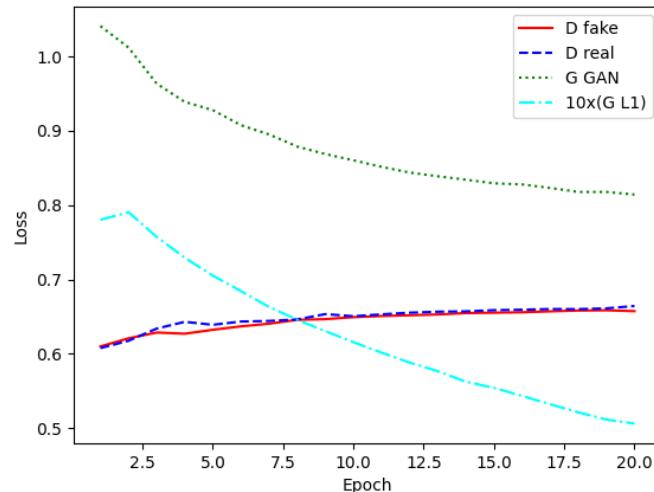
Hình 3.5: Giá trị mất mát của bộ sinh trên hai tập huấn luyện & xác thực qua từng epoch

### 3.4.5 Huấn luyện mạng GAN với bộ sinh ResNet

Tải các thông số của bộ sinh đã được tiền huấn luyện độc lập trước đó, sau đó đưa vào để huấn luyện chung với bộ phân biệt với hệ số cân bằng  $\lambda = 100$ . Thuật toán tối ưu vẫn sẽ dùng là Adam để tối ưu bộ phân biệt ( $D$ ) và bộ sinh ( $G$ ) với các thông số được chọn là:

$$\alpha = 2 \times 10^{-4}, \beta = (0.5, 0.999), \epsilon = 10^{-8}$$

Cho chạy với batch size là 16 và 20 epoch. Mỗi epoch mất khoảng 10-16 phút (chưa tính phần xác thực).



Hình 3.6: Giá trị các thành phần mất mát của bộ phân biệt và bộ sinh qua từng epoch

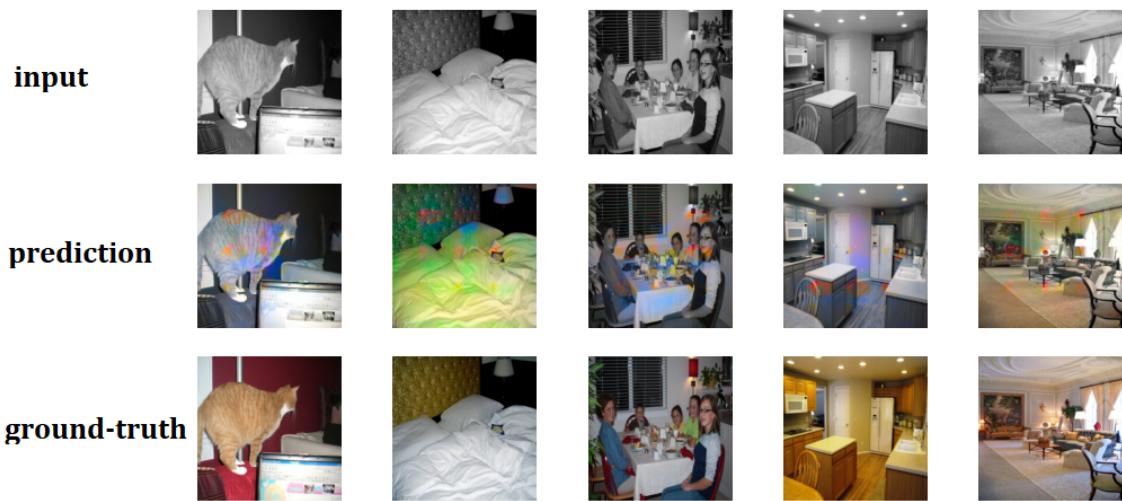
Nhận thấy xu hướng của giá trị mất mát khi huấn luyện mô hình GAN là giá trị mất mát của bộ phân biệt tăng thì giá trị mất mát của bộ sinh sẽ giảm và cả hai *dường như* (chưa qua kiểm tra thực nghiệm) đều sẽ hội tụ về một giá trị. Lý giải điều này, khi bộ phân biệt đạt đến một người thông minh nhất định, bộ sinh thì lại càng thông minh qua mỗi epoch hiển nhiên sẽ đánh lừa (*fool*) 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 khi xem xét một vài kết quả của mô hình sau mỗi epoch,<sup>13</sup> em quyết định chọn mô hình thứ 15 (tức sau khi huấn luyện qua 15 epoch).

<sup>13</sup>Sẽ được trình bày trong phần 4 **Thử nghiệm mô hình**

## 4 Thủ nghiệm mô hình

### 4.1 Mô hình GAN với bộ sinh đơn giản kết hợp chuẩn 1



Hình 4.1: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản kết hợp chuẩn 1 trên dữ liệu xác thực của tập dữ liệu **COCO** sau epoch thứ 34

Cơ bản mà nói, việc huấn luyện mô hình là khả thi khi mô hình cũng đã có thể tò được màu sau vài chục epoch. Nhưng giới hạn về phần cứng cũng như là tập dữ liệu không đủ nhiều và tốt để có thể huấn luyện được mô hình một cách trọn vẹn.

Bản thân việc huấn luyện một mạng GAN không hề dễ dàng khi GAN rất nhạy cảm với những **thông số cấu trúc** (*hyperparameter*). Sau nhiều lần thử nghiệm thất bại, em cũng đã rút ra một vài kinh nghiệm khi huấn luyện GAN. Một trong số đó như là việc trong hầu hết các mô hình GAN, ta có thể khởi tạo các **trọng số** (*weight*) và bias theo phân phối chuẩn  $\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0, 1)$  để quá trình huấn luyện có kết quả được nhanh hơn.

Do những hạn chế trên, em đã chuyển phương hướng thay vì xây dựng một bộ sinh từ đầu thì sẽ tận dụng khả năng học chuyển tiếp để tận dụng những mô hình có sẵn.

### 4.2 Mô hình GAN với bộ sinh ResNet kết hợp chuẩn 1



Hình 4.2: Kết quả dự đoán của mô hình GAN với bộ sinh ResNet kết hợp chuẩn 1 trên dữ liệu xác thực của tập dữ liệu **COCO** sau epoch thứ 15

Đây là mô hình được huấn luyện dựa vào bộ sinh đã được tiền huấn luyện độc lập trước với 20 epoch, sau đó kết hợp với huấn luyện đối nghịch với bộ phân biệt  $70 \times 70$  sau 15 epoch.

Kết quả cho thấy rằng mô hình sau khi huấn luyện đã có khả năng tô màu cho các đối tượng trong ảnh, màu ít hoặc gần như không bị lem. Mô hình cũng không dự đoán chính xác tuyệt đối màu của một số đối tượng nhưng hợp lý, theo đúng hướng giải quyết vấn đề mà ta mong muốn.

Một vài vùng ảnh, mô hình có vẻ như vẫn chưa thực sự tô được màu khi để màu gần giống với mức xám ban đầu. Để giải cho hạn chế này, có thể những vùng chưa được tô màu một cách hợp lý là trong quá trình huấn luyện, mô hình chưa được học qua ảnh có bố cục, đối tượng tương tự. Điều này dẫn đến một hạn chế khác nữa là đối với những ảnh không có bố cục rõ ràng (ảnh logo), mô hình hoàn toàn có thể không có khả năng tô được màu vì không nhận diện đối bối cảnh.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 hình ảnh hai cô gái cầm vợt tennis (ảnh thứ 2 từ phía bên phải), phần dưới của áo được tô màu hồng, còn phần trên của áo lại màu xanh. Ở đây, nếu tô cho phần áo màu hồng hay màu xanh thì đều chấp nhận được, nhưng mô hình lại chọn cách tô hai phía của áo, do được chia bởi phần cách tay, bởi hai màu khác nhau.



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

Những hạn chế trên, ta hoàn toàn có thể khắc phục bằng cách cho mô hình được huấn luyện thêm với những tập dữ liệu khác. Một hướng khác, ta có thể sử dụng một mạng được huấn luyện trước tốt hơn **ResNet18** như **ResNet50**, **ResNet101** hoặc **DenseNet** khi giới hạn phần cứng cho phép.



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

**Hình 4.4** thực sự là một kết quả bất ngờ mà hiện tại em vẫn chưa lí giải được tại sao mô hình cho kết quả không thực sự tốt. Việc mô hình không có khả năng làm việc với ảnh mặt người là không khả thi khi ở phần **4.3 So sánh bộ sinh ResNet trước và khi được huấn luyện đối nghịch** cũng có một ảnh mặt người tương tự và mô hình vẫn cho kết quả ổn. Đây là một trường hợp đặc biệt (*edge case*) cần thêm nhiều tìm hiểu.

#### 4.3 So sánh bộ sinh ResNet trước và khi được huấn luyện đối nghịch

Việc so sánh giữa hai mô hình này sẽ cho ta cái nhìn về sự ảnh hưởng của mô hình sau khi được huấn luyện đối nghịch. Cụ thể, em đã chủ động lựa chọn ngẫu nhiên một vài (số lượng 5) tấm ảnh từ Google về ảnh 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 được lựa chọn để thử nghiệm sự khác biệt

Trước khi đưa cho mô hình thực hiện tô màu, không khó để đoán được rằng hình ảnh đầu tiên (phía trái ngoài cùng) có ít chi tiết nhất nên sẽ có kết quả tốt nhất. Ngược lại, hình ảnh cuối cùng (phía phải ngoài cùng) có rất nhiều chi tiết nên kết quả khó có thể 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 4.6: So sánh kết quả dự đoán của hai mô hình là mô hình chỉ dùng chuẩn 1 (phía trên) với mô hình GAN kết hợp chuẩn 1 (phía dưới)

Ta có thể thấy trong **Hình 4.5** rằng mô hình sau khi được huấn luyện theo kiểu GAN cho kết quả có màu tươi sáng, chân thật hơn so với khi ta chỉ dùng chuẩn 1 để huấn luyện. Điều này là do mô hình dùng chuẩn 1 nếu không tìm được màu phù hợp thì sẽ sử dụng giá trị trung bình, tức là giá trị  $(\mathbf{a}, \mathbf{b}) \approx (0, 0)$  vì  $-1 \leq \mathbf{a}, \mathbf{b} \leq 1$ . Khi đó, màu của điểm ảnh sẽ không khác so với mức xám  $\mathbf{L}$  đầu vào là bao, khiến cho kết quả của mô hình chỉ dùng chuẩn 1 dường như chỉ là thay đổi độ sáng của ảnh đầu vào chứ không thực sự tô được màu. Và hạn chế này đã được khắc phục rất nhiều khi ta đưa mô hình qua huấn luyện đối nghịch.

## 5 Kết luận & Hướng phát triển

### 5.1 Kết luận

Về kết quả đề tài, em đã thành công trong việc xây dựng một mô hình có khả năng **tự động tô màu** những ảnh xám, **cũ** một cách **chân thực, hợp lý**. Kiến trúc mô hình là kiến trúc của một **mạng đối nghịch tạo sinh** với **bộ phân biệt**  $70 \times 70$  kết hợp với **bộ sinh theo kiến trúc mạng Unet** được tạo nên từ phương pháp **học chuyển tiếp** từ **mạng ResNet18**.

Dẫu vậy, mô hình vẫn chưa có kết quả tốt khi áp dụng với những ảnh có bối cảnh, đối tượng mà mô hình chưa được học trong quá trình huấn luyện. Như đã đề cập ở phần 4 **Thử nghiệm mô hình**, ta hoàn toàn có thể cải thiện trí tuệ mô hình thông qua việc tìm thêm nhiều tập dữ liệu với các bối cảnh, đối tượng phong phú để huấn luyện thêm hoặc cải thiện bộ sinh bằng cách lựa chọn một mạng có kết nối tinh tế hơn ResNet18. Một cách tiếp cận khác để cải thiện mô hình đó là huấn luyện nhiều bộ sinh, mỗi bộ sinh sẽ đảm nhiệm mỗi chủ đề khác nhau, như vậy chỉ cần cho bộ sinh học những tấm ảnh liên quan đến chủ đề của mình. Bằng việc chuyên môn hóa cho chủ đề của bộ sinh, ta không còn yêu cầu một bộ sinh có thể tô màu tốt cho mọi bối cảnh tuy nhiên lại giúp quá trình huấn luyện được đơn giản hơn do không phải học quá nhiều và sẽ cho kết quả tốt hơn cả khi được đưa cho ảnh đúng chuyên môn.

So với mục tiêu đề ra trong phần **1.2 Mục tiêu**, khi xây dựng một mô hình dự đoán được màu của những đối tượng trong ảnh xám một cách hợp lý, chất lượng không yêu cầu phải tốt như làm thủ công, giúp tiết kiệm thời gian trong quá trình phục hồi màu cho ảnh xám thì mô hình đã xử lí được vấn đề được nêu ra.

Sau quá trình tìm hiểu và hoàn thành đề tài, bản thân em đã học thêm được nhiều kiến thức về những khía cạnh màu của ảnh số ngày nay (RGB, Lab). Thêm vào đó, nhờ việc thực sự đắm mình vào học sâu (*diving into deep learning*), em đã có một cái nhìn rõ ràng hơn về học máy/học sâu trong một mảng khổng lồ như là **trí tuệ nhân tạo** (*artificial intelligence*).

## 5.2 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. Kết hợp với tinh chỉnh bằng phương pháp thủ công, ta hoàn toàn có thể có những kết quả phục chế đẹp đẽ và cũng nhanh chóng.

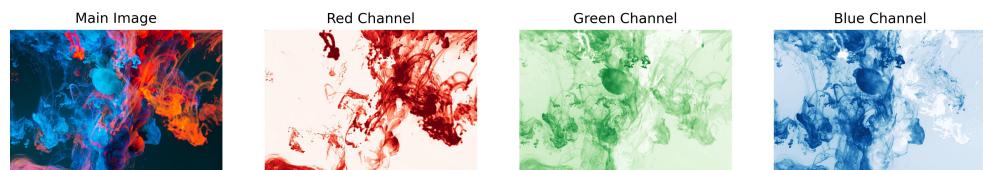
Không chỉ dừng lại ở hình ảnh, ta cũng có thể áp dụng mô hình để phục chế màu cho các video, vì bản chất một video là sự kết hợp từ nhiều khung hình (*frame*) ghép lại. Trong thư mục `test_results`<sup>14</sup> có kết quả thử nghiệm của mô hình qua việc phục chế màu một đoạn nhỏ về một phóng sự về chiến tranh ở Việt Nam.



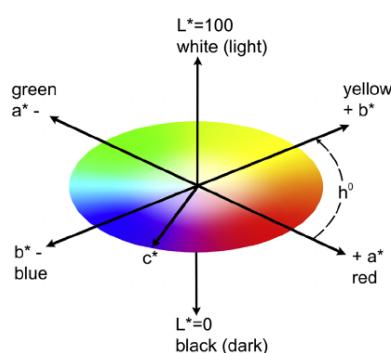
Hình 5.1: Ảnh Trường DH Bách Khoa TP.HCM ngày xưa (trái) và sau khi tô màu (phải)

<sup>14</sup> Được chứa trong thư mục [https://github.com/dee-ex/EE3151\\_SEM202\\_PROJECT](https://github.com/dee-ex/EE3151_SEM202_PROJECT)

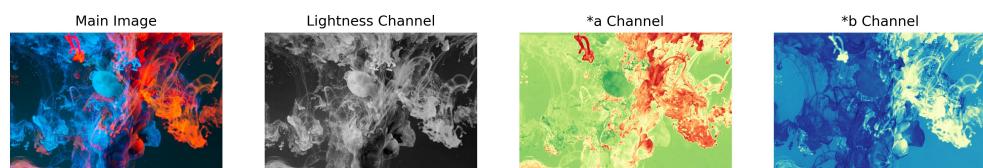
## 6 Phục lục - Ảnh màu



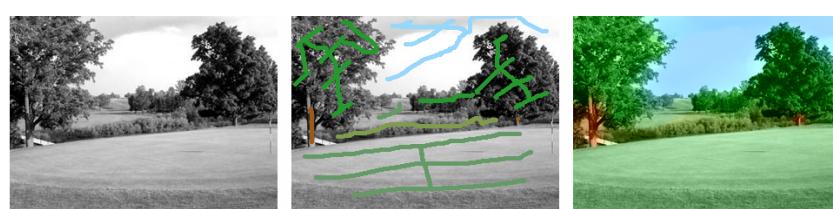
Hình 2.1: Các kênh màu đỏ, lục và lam được tách ra riêng biệt. Nguồn: <https://bit.ly/3eKaaBj>



Hình 2.2: Không gian màu Lab. Nguồn: <https://bit.ly/3nFIQIp>



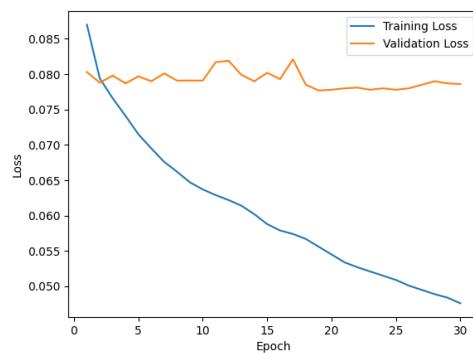
Hình 2.3: Các giá trị **L**, **a** và **b** được tách ra riêng biệt. Nguồn: <https://bit.ly/3eKaaBj>



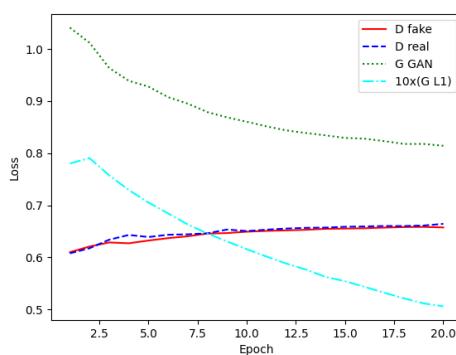
Hình 2.4: Mô tả ý tưởng thuật toán đánh dấu vài điểm ảnh. Nguồn: <https://bit.ly/3eKdmwN>



Hình 2.5: Mô tả ý tưởng thuật toán ảnh có bộ cục tương tự. Nguồn: <https://bit.ly/3eKdmwN>



Hình 3.5: Giá trị mất mát của bộ sinh trên hai tập huấn luyện & xác thực qua từng epoch



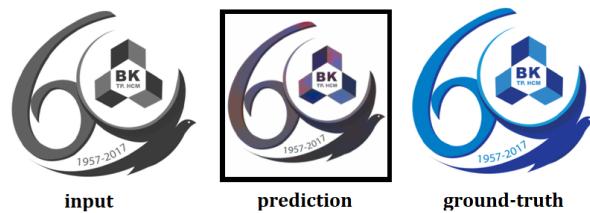
Hình 3.6: Giá trị các thành phần mất mát của của bộ phân biệt và bộ sinh qua từng epoch



Hình 4.1: Kết quả dự đoán của mô hình GAN với bộ sinh đơn giản kết hợp chuẩn 1 trên dữ liệu xác thực của tập dữ liệu **COCO** sau epoch thứ 34



Hình 4.2: Kết quả dự đoán của mô hình GAN với bộ sinh ResNet kết hợp chuẩn 1 trên dữ liệu xác thực của tập dữ liệu **COCO** sau epoch thứ 15



Hình 4.3: Mô hình hoàn toàn bê tắc 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 bộ cục đơn giản



Hình 4.6: So sánh kết quả dự đoán của hai mô hình là mô hình chỉ dùng chuẩn 1 (phía trên) với mô hình GAN kết hợp chuẩn 1 (phía dưới)



Hình 5.1: Ảnh Trường DH Bách Khoa TP.HCM ngày xưa (trái) và sau khi tô màu (phải)

## Tài liệu tham khảo

- [1] Aladdin Persson, “Building our first simple GAN”, <https://youtu.be/OljTVUVzPpM>, (2020).
- [2] Aladdin Persson, “Pix2Pix Paper Walkthrough”, <https://youtu.be/SuddDSqGRzg>, (2021).
- [3] Aladdin Persson, “Pix2Pix implementation from scratch”, <https://youtu.be/SuddDSqGRzg>, (2021).
- [4] Aladdin Persson, “U-NET Paper Walkthrough”, <https://youtu.be/oLvmLJkmXuc>, (2021).
- [5] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, “Dive into Deep Learning”, (2021).
- [6] Christopher Thomas BSc Hons. MIAP, “U-Nets with ResNet Encoders and cross connections”, <https://bit.ly/3ttEu8x>, (2019).
- [7] Connor Shorten, “DCGANs (Deep Convolutional Generative Adversarial Networks)”, <https://bit.ly/3nFC7OC>, (2018).
- [8] Connor Shorten, “Pix2Pix”, <https://bit.ly/3eMDGGt>, (2019).
- [9] Eli Stevens, Luca Antiga, Thomas Viehmann and Foreword by Soumith Chintala, “Deep Learning with PyTorch”, Manning Publications, New York (2020).
- [10] Emil Wallner, “How to colorize black & white photos with just 100 lines of neural network code”, <https://bit.ly/3u8uSkN>, (2017).
- [11] Harshall Lamba, “Understanding Semantic Segmentation with UNET”, <https://bit.ly/3tdDK7B>, (2019).
- [12] Ian J. Goodfellow et al. “Generative Adversarial Networks”, <https://arxiv.org/abs/1406.2661>, (2014).
- [13] Jason Brownlee, “A Gentle Introduction to Pix2Pix Generative Adversarial Network”, <https://bit.ly/2RhbbsC>, (2019).
- [14] Joseph Rocca, “Understanding Generative Adversarial Networks (GANs)”, <https://bit.ly/3vzdPIU>, (2019).
- [15] Juan, “What range does skimage use in LAB color space for each channel?”, <https://bit.ly/3eaonbt>, (2020).
- [16] Mehdi Mirza & Simon Osindero, “Conditional Generative Adversarial Nets”, <https://arxiv.org/abs/1411.1784>, (2014).
- [17] Moein Shariatnia, “Colorizing black & white images with U-Net and conditional GAN — A Tutorial”, <https://bit.ly/3eKaaBj>, (2020).
- [18] Olaf Ronneberger, Philipp Fischer & Thomas Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, <https://arxiv.org/abs/1505.04597>, (2015).
- [19] Pham Dinh Khanh, “Image Segmentation”, <https://bit.ly/335IEZE>, (2020).
- [20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou & Alexei A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks”, <https://arxiv.org/abs/1611.07004>, (2016).
- [21] Richard Zhang, Phillip Isola & Alexei A. Efros, “Colorful Image Colorization”, <https://arxiv.org/abs/1603.08511>, (2016).
- [22] Sahil, “Understanding PatchGAN”, <https://bit.ly/3xJy0Wo>, (2020).
- [23] Trung Thanh Nguyen, “Auto Colorize Grayed Image With Deep Learning”, <https://bit.ly/3theTj4>, (2018).
- [24] Tuan Nguyen, “Conditional GAN (cGAN)”, <https://bit.ly/3udFFKH>, (2020).
- [25] Tuan Nguyen, “Deep Convolutional GAN (DCGAN)”, <https://bit.ly/3uhxtcb>, (2020).
- [26] Tuan Nguyen, “Giới thiệu về GAN”, <https://bit.ly/3aXhHvD>, (2019).
- [27] Tuan Nguyen, “Pix2pix”, <https://bit.ly/3xGVR9h>, (2020).
- [28] Wikipedia, “CIELAB color space”, [https://en.wikipedia.org/wiki/CIELAB\\_color\\_space](https://en.wikipedia.org/wiki/CIELAB_color_space), (2021).